Supreme Court, U. S.
FILED
JAN 22 1980
MICHAEL RODAK, JR., CLERK

IN THE

Supreme Court of the United States

OCTOBER TERM, 1979

No. 79-855

SIDNEY A. DIAMOND, COMMISSIONER
OF PATENTS AND TRADEMARKS

Petitioner,

v.

JOHN J. BRADLEY AND BENJAMIN S. FRANKLIN

Respondents.

BRIEF IN OPPOSITION OF A WRIT OF CERTIORARI TO THE UNITED STATES COURT OF CUSTOMS AND PATENT APPEALS

Of Counsel:

FAITH F. DRISCOLL
HENRY L. HANSON
RONALD T. REILING

NICHOLAS PRASINOS
Honeywell Information Systems, Inc.
Office of the General Counsel
200 Smith Street
Waltham, Massachusetts 02154
(617) 895-6129

Counsel for Respondents

Washington, D.C. • CLB PUBLISHERS' • LAW PRINTING CO. • (202) 393-0625

TABLE OF CONTENTS

Page

INTRODUCTION 1

COUNTERSTATEMENT OF QUESTION PRESENTED 5

CONSTITUTIONAL AND STATUTORY PROVISIONS INVOLVED 5

STATEMENT OF THE CASE 6

ARGUMENT 10

SUMMARY 19

CONCLUSION 21

APPENDIX A 1a

APPENDIX B 1b

APPENDIX C 1c

TABLE OF AUTHORITIES

Cases:

Bradley and Franklin, In re, 600 F.2d 807 (1979) 9

Chatfield, In re, 545 F.2d 152 (1976) 4

de Castelet, In re, 562 F.2d 1236 (1977) 18

Diehr, In re, 602 F.2d 982 (1979) 18

Eibel Process Co. v. Minnesota & Ontario Paper Co., 261 U.S. 45 (1923) 13,16

Funk Brothers Seed Co. v. Kalo Innoculant Co., 333 U.S. 127 (1948) 11

Gelnovatch, In re, 595 F.2d 32 (1979) 18

Gottschalk v. Benson, In re, 409 U.S. 63 (1972) ... 2,3,9,11,19

Graham v. Deere, 383 U.S. 1 (1966) 13,16

Johnson, Parrack and Lunsford, In re, 589 F.2d 1070 (1978) 4,18

Mackay Radio & Telegraph Co. v. Radio Corporation of America, 306 U.S. 86 (1939) 11,16

Maucrops, In re, No. 79-554 (CCPA, November 1, 1979) 17

Parker v. Flook, 437 U.S. 584, 98 S.Ct. 2522 (1978) ..... *passim*

Richman, In re, 563 F.2d 1026 (1977) 18

Safety Car Heating & Lighting Co. Inc, v. General Electric Co. et al 155F.2d 937 (1946) 15

Sarkar, In re, 588 F.2d 1330 (1978) 18

U.S. v. Adams, 383 U.S. 39 (1966) 14,16

Waldbaum, In re, 559 F.2d 611 (1977) 18

Statutes:

35 USC 101 ..... *passim*

35 USC 112 12,17

Miscellaneous:

Rules of Practice in Patent Cases 12

Computer Dictionary and Handbook (2d ed. 1972), C. J. Sippl and C. P. Sippl 8,19

IN THE

Supreme Court of the United States

OCTOBER TERM, 1979

No. 79-855

SIDNEY A. DIAMOND, COMMISSIONER
OF PATENTS AND TRADEMARKS

Petitioner,

v.

JOHN J. BRADLEY AND BENJAMIN S. FRANKLIN

Respondents.

BRIEF IN OPPOSITION OF A WRIT OF CERTIORARI TO THE UNITED STATES COURT OF CUSTOMS AND PATENT APPEALS

INTRODUCTION

There is no basis for issuance of a Writ of Certiorari in this case. None of the criteria which normally are considered in the Court's discretion are present here.[1] In particular, there is no conflict with any applicable decision of this Court.

[1] See Supreme Court Rule 19.

The machine invention which the Respondents seek to patent has been mistakenly represented by the Petitioner to be a computer program when in fact it is clearly not a computer program.

The Respondents strongly submit that whether computer programs per se are patentable is not an issue in this Petition. It should be noted that the real party in interest, the Assignee of the Respondents' Application, has been on record before this Court as opposing the patentability of computer programs.[2] The Respondents' invention is not a computer program; if it were such a program, no patent application would have been filed. Furthermore, Respondents and others are of the view that computer programs are more properly the subject matter of copyright.[3]

The invention claimed by Respondents is a new computer machine (R6-111 and R207-208).[4] The United States Court of Customs and Patent Appeals unanimously found that the invention was a machine (PW-15a).[5]

---

[2] See Brief of Honeywell Inc. Amicus Curiae in Gottschalk v. Benson, 409 US 63, (1972).

[3] See Final Report of the National Commission on New Technological Uses of Copyright Works. The Commission on page 1 has recommended to President Carter and to the U.S. Congress that:

> "The new copyright law should be amended: (1) to make it explicit that computer programs, to the extent that they embody an author's original creation are proper subject matter of copyright. . . ."

[4] The letter R followed by a numeral denotes the page number of the Transcript of the Record before the U.S. Court of Customs and Patent Appeals to which reference is made.

[5] PW indicates the page number of the Petition for Writ of Certiorari that the Petitioner has filed before this Court.

Even the Petitioner through the Associate Solicitor of the United States Patent and Trademark Office has admitted that Respondents' invention is a machine.[6] With the admission that the Respondents' invention is a machine, the issue of statutory subject matter is resolved because all the rejections and objections of the Patent and / Trademark Office have been overcome.[7] Under 35 USC 101 new machines are explicitly stated to be statutory subject matter. Hence, there is no real issue for this Court to decide.

It is submitted that by filing the Petition for Writ of Certiorari, the Commissioner of Patents and Trademarks is misleading this Court by "labeling" the invention a computer program.[8] The tactic of labeling inventions as

---

[6] See Appendix A.

[7] The question of "newness" had already been addressed in prosecution and resolved in Respondent's favor. (See PW-34a). There the Examiner cited the Hauck patent as prior art which was successfully rebutted by the Respondents and the rejection withdrawn by the Examiner with only a 35 USC 101 rejection remaining. (PW-37a and 41a). On the record before this Court, the Respondents' invention is acknowledged to be a new machine as claimed.

[8] Even if the invention were a program, which it is not, this Court has consistently refused to invade the legislative prerogative of Congress with regard to such matters. In Gottschalk v. Benson, 409 US 63, (1972) this Court stated:

> "If these programs are to be patentable, considerable problems are raised which only committees of Congress can manage, for broad powers of investigation are needed, including hearings which canvass the wide variety of views which those operating in this field entertain. The technological problems tendered in the many briefs before us indicate to us that considered action by the Congress is needed."

(continued)

computer programs and then finding such inventions unpatentable has long been followed by the Patent and Trademark Office. The United States Court of Customs and Patent Appeals has repeatedly admonished the Patent and Trademark Office about its use of this tactic.[9]

(footnote continued from preceding page)

In Parker v. Flook, 437 US 198 (1978) this Court again reiterated its position as follows:

> "Difficult questions of policy concerning the kinds of programs that may be appropriate for patent protection and the form and duration of such protection can be answered by Congress on the basis of current empirical data not easily available to this tribunal."

[9]The Court of Customs and Patent Appeals in In re Chatfield, 545 F.2d 152, 191 USPQ 730 (1976) stated:

> "The unique technological aspects of digital computers and the changes wrought in their operation by associated software programs, coupled with the limited number of precedental court decisions, illustrate the undesirability of deciding the issue of patentability by mere use of the label 'computer program.'"

In re Johnson, Parrack and Lunsford 589 F.2d 1070, 200 USPQ 199 (1978) the Court of Customs and Patent Appeals said:

> "The Solicitor argues in his supplemental brief that the claims at issue are not statutory subject matter because they are computer programs and computer programs are not patentable under Benson and Flook. This broad statement is not at all germane to the considerations before this Court and is an erroneous statement of the law. We stated in In re Chatfield, supra at 155, 191 USPQ 733 that 'the mere labeling of an invention as a computer program does not aid in decision making and the Supreme Court declined to decide either Benson or Flook on such broad nonsubstantive ground. . . .'"

COUNTERSTATEMENT OF QUESTION PRESENTED

The "Question Presented" by Petitioner reflects a clear misunderstanding of the invention and a misconstruction of the record.[10] It is based upon a false premise. The essential issue in the present case on which Petitioner seeks this Court's intervention may be accurately and properly stated as follows:

> "Is a machine invention which is in the technological arts, which is constructed and claimed as a combination of elements, patentable subject matter under 35 USC 101?"

CONSTITUTIONAL AND STATUTORY PROVISIONS INVOLVED

Article I, Section 8, Clause 8, of the Constitution of the United States provides:

> The Congress shall have Power * * *
>
> * * *
>
> To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries.* * *

Petitioner has correctly stated the applicable statutory provision, 35 USC 101.

---

[10]The Examiner, the Board of Appeals and the Solicitor of the U.S. Patent and Trademark Office have repeatedly misinterpreted and misconstrued the Respondents' invention. See Appendix B for illustrative errors.

STATEMENT OF THE CASE

The United States Court of Customs and Patent Appeals correctly describes the inventive subject matter and accurately relates the reasoning underlying the application of 35 USC 101 to this inventive subject matter.[11] Respondents need not, therefore, repeat here these details, except in the most fundamental way so that a clear and undistorted picture of the nature of the invention and its judicial history is presented.

This invention is one of a series of inventions which collectively define an entirely new computer machine which is now being marketed worldwide as the Honeywell Series 60, Level 64 computer product line. The invention, as disclosed and claimed in the Respondents' application, is directed to a unique design of a portion of this Level 64 computer system. This portion which the Respondents have invented is a new and useful computer machine.[12] The new computer machine utilizes as part of this unique design, an implementation of the invention which includes firmware as one element of the new combination. This invention can also be implemented totally in a hardwired embodiment—that is without any firmware element. It is merely a design implementation choice. This type of invention if imple-

---

[11] See PW 1a-15a.

[12] A complete discription of the invention from the Record below (R3-R111) is contained in Appendix C (pps. 1c-111c) The description includes twenty-two (22) sheets of drawings, eighty-six (86) pages of specification and six (6) claims. The Petitioner in his Petition and Brief has offered, out of context, a portion of the description of the invention compromising only one (1) sheet of drawing and five (5) pages of description (PW 42a-46a) purporting to describe this complex invention.

mented totally in hardwire would clearly be statutory subject matter under 35 USC 101.[13] The invention cannot be implemented as a computer program to be executed by any general purpose computer. It is not a computer program invention.

Perhaps the nature of the invention and how it is completely different from a computer program may best be illustrated by a simple analogy. A consideration of a player piano and its operation is particularly useful in pinpointing and understanding the nature of the invention.

Player pianos and their method of operation, using rolls into which the notes of a musical composition have been punched, are well known. A player piano, once designed and operative, will respond to a musical composition encoded in a roll and fed into it by automatically playing the composition in the manner intended by the composer. The piano, once the implementation has been fixed, will always be the same combination of elements and operate in the same manner. However, the internal mechanism of a player piano may obviously be redesigned and implemented as a new combination which operates in a new and different manner. The redesigned piano is a new player piano (a new combination). Externally the new combination still performs the musical compositions recorded

[13]The following U.S. Patents are examples of hardwired implementations of similar types of inventions classified in the same technical class 340 and 172.5 as the Respondents' inventions: 3,454,933; 3,454,931; 3,425,039; 3,323,110; 3,522,589; 3,344,404; and 3,564,504. Additionally, the following U.S. Patents involve similar types of inventions implemented in firmware: 3,400,371; and 4,028,670.

in any piano roll inserted therein.

By analogy therefore the invention the Respondents are claiming is a new piano (new computer machine). They are not claiming a new piano roll (computer program). The implementation of the new internal hardware redesign is a new combination, an apparatus which is a new machine which clearly passes through the gates of 35 USC 101.

There are two separate individuals involved in the conception of the two separate products involved in the foregoing analogy. First, the composer (programmer) and second, the piano (machine) designer. While both may be very creative in their respective fields, their roles are different and distinct and the products they have created must be considered separately in terms of the application of the patent, copyright or other laws of the United States thereto.

The Respondents' computer invention is the work product of computer machine designers, (equivalent to the piano designer in the analogy). They have designed a new computer machine which operates more efficiently.

The inventors in conceiving this new machine have drawn from their creative ability and computer expertise. In implementing their invention, the inventors have utilized the latest hardware technology available which includes firmware.[14] Once the invention is incorporated into the computer machine, any compatible program

[14]The Computer Dictionary and Handbook, by Charles J. Sippl and Charles P. Sippl, published by Howard W. Sams & Son, Inc., N.Y., defines "firmware" on page 186 as follows: "Firmware—logic circuits in read-only memory. . . ."

(old or new) may be used in the new, improved computer machine. A programmer using the computer machine or writing programs therefor is normally not aware and does not have to know, how the computer machine is constructed or operates internally.

Respondents are applying for a patent covering the new combination, the new machine (new piano). They are not claiming, contrary to what the Petitioner is alleging, a computer program (piano roll). Furthermore, they are not contending that the running of a program (old or new) on the new, improved computer machine results in still another new machine; this is not under consideration here.

What the inventors have conceived is a new combination of hardware elements, which includes a firmware element. Accordingly, when any program such as a program for translating "texts from Russian to English or for determining a hitherto unknown and faster route between Washington and New York"[15] is run on this new, improved computer machine, it can be run more efficiently.

The Respondents' apparatus combination Claims 1-6 were finally rejected by the Examiner under 35 USC 101 as unpatentable and the rejection was "grounded on an erroneous interpretation of the Supreme Court's decision and opinion in *Benson*, namely, that all computer programs or program related inventions are nonstatutory under Section 101."[16] The Board of

[15]PW-12. (See also Appendix B, page 36).

[16]In re Bradley and Franklin, 600 F.2d 807, 202 USPQ 480 at 484 (1979).

Appeals affirmed the Examiner (R203) further considering the invention in view of *Parker v. Flook* (R197), 437 US 188, (1978).

The United States Court of Customs and Patent Appeals, on July 5, 1979, unanimously reversed both the Examiner and the Board of Appeals stating:

> ". . . We disapprove the Board's distillation of Appellants' claim down to the information contained in the firmware in order to hold it nonstatutory."
>
> "We find that the invention is a combination of tangible hardware elements—a machine. . . ."
>
> ". . . we do not find the invention to be nonstatutory under the authority of *Benson* or *Flook*, or under the authority of our own cases. . . ."

ARGUMENT

There is no basis for granting the petition. The decision of the court below[17] does not conflict with any decisions of this Court; it is fully in conformance with the applicable Constitutional and Statutory provisions; and, it in no way extends the scope of the patent laws.

The Petitioner relies heavily upon this Court's decision in *Parker v. Flook*, 437 US 584 (1978). It is submitted that the *Flook* decision is not even relevant to the invention under consideration here for the following reasons. First of all the Petitioner erroneously characterizes the Respondents' invention as an algorithm.[18] This Court in *Flook* and *Benson* defined an algorithm as follows:

[17]PW 1a-15a.

[18]PW 10.

> "We use the word 'algorithm' in this case as we did in *Gottschalk v. Benson*, 409 U.S. 63, 65, 175 USPQ 673, 674 to mean '(a) procedure for solving a given type of mathematical problem. . . .' "[19]

No algorithm is disclosed in the Respondents' specification, no algorithm is claimed and no mathematical problem is to be solved.

In addition, this Court in its 6-3 decision explicitly held in *Flook*:

> "Very simply, our holding today is that a claim for an improved method of calculation, even when tied to a specific end use, is unpatentable subject matter under Section 101."[20]

The Respondents' invention is a machine and no method of calculation is claimed. The Petitioner has obviously misread the *Flook* decision and its applicability to the invention involved here.

Even if the *Flook* decision were applicable, the Petitioner has chosen to interpret this Court's opinion in a very questionable manner. The Court stated:

> "Mackay Radio and Funk Brothers point to the proper analysis for this case: The process itself, not merely the mathematical algorithm, must be new and useful. Indeed, the novelty of the mathematical algorithm is not a determining factor at all."[21]

However, the Petitioner confusingly alleges a two-step analysis is required under 35 USC 101 for a "patent claim involving an algorithm." The Respondents do not claim an algorithm. Therefore, the Petitioner's alleged

---

[19] 98 S. Ct. at 2523 (footnotes).

[20] *Parker v. Flook*, 98 S. Ct. at 2528 (footnotes).

[21] Ibid (2526).

two-step analysis is not even applicable to the Respondent's machine invention.

Going even further if the Petitioner's alleged two-step analysis were to be applied to Respondents' invention, the result of such analysis is that the invention is nonetheless statutory subject matter uder 35 USC 101. The first step of Petitioner's alleged test is that one must determine whether the claim contains a principle, formula, idea or concept which is unpatentable which then must be separated from the rest of the claim. There is no such principle, formula, idea or concept in Respondents' claim. Thus it passes the first step. The second step of Retitioner's alleged test is to determine if what remains of the claim is old in the art. The Respondents have continuously maintained that what they have invented and what is being claimed is a new combination of elements, and this has never been successfully rebutted. The Petitioner is now alleging that everything in the claim is old without any support in the Record for the allegation.[22] Thus even by the

[22]The Commissioner of the Patent and Trademark Office has for the first time in the prosecution of this invention, improperly and belatedly referred to, in the Petition, the fact of common subject matter in Respondents' application and the application of Carre. (PW 8-9 footnotes). The issue is not before this Court. However, Respondents note that it is accepted practice for inventors working on a common machine to disclose their separate and distinct inventions (which are both different parts of the overall machine), by describing the total machine. Accordingly, the two patent applications include common subject matter. In view of the Patent and Trademark Office Rule 71 and 35 USC 112 of the Patent Laws, the two inventions, probably could not properly be disclosed without including common subject matter. The reference to common subject matter in the Petition raises a question of novelty which is not before this Court. The issue was raised by the Examiner during prosecution and decided in Respondents' favor. (See p. 3 supra).

application of the Petitioner's own alleged two-step analysis, the Respondents' invention is patentable subject matter under 35 USC 101.

This Court in *Flook* expressly reiterated the long standing and legally sound principle that an invention must be considered as a whole[23] and then proceeded to state:

> "Respondents' process is unpatentable under Paragraph 101 not because it contains a mathematical algorithm as one component, but because once the algorithm is assumed to be within the prior art, the application considered as a whole contains no patentable invention."[24]

In *Flook* the Respondents did not deny that the invention was a process, that the process was old, and that the novelty lay only in a claimed mathematical algorithm. It was thus established by the Court that the sequence of steps in that particular process under

---

[23]Parker v. Flook 96 S. Ct. at p. 2526.

[24]This requirement was also stated by this Court in Graham v. Deere, 383 US 1 (1966), when it noted that the invention as a whole must be viewed in determining patentability, and that the whole encompasses both conception and implementation.

This reasoning is also consistent with this Court's opinion in Eibel Process Co. v. Minnesota & Ontario Paper Co., 261 US at 45 (1923). There the patentee discovered that wrinkling and malformation of paper in a paper making machine occurred because the stock did not flow fast enough at a critical point in the process. Eibel simply increased the pitch of the path of the stock to take advantage of the force of gravity to increase the flow rate. This Court held the patent valid deciding that the patentee had made a patentable contribution. It rejected arguments that the patentee was claiming a force of gravity, and considered the invention as a whole in determining patentability.

consideration was also old. It was obvious that only one step of the process was alleged to be novel. When in accordance with this Court's reasoning the mathematical algorithm (the one step alleged to be novel) was also assumed to be within the prior art (i.e., old), then that specific process as a whole was also old. Accordingly, there was no patentable subject matter contained by the claimed process.

The Respondents' invention is vastly different than the invention considered by the Court in *Flook*.

To illustrate the error of Petitioner's reasoning by analogy, assume that one is considering the following old elements: (1) wheels, (2) shafts, (3) rope and (4) a frame.

A first combination which can be claimed is a cart that is constructed with the wheels, shaft and frame to be pulled along by the rope.

A second claimable combination is a system of pulleys constructed with the wheels, shaft, frame and rope to lift large weights.

A third combination is a steering mechanism constructed with the wheels, shaft, frame and rope to steer an airplane or a boat.

All of these combinations have the same well known elements. What gives uniqueness to each combination is the arrangement of these known elements with respect to each other and how they cooperate with each other, i.e., how they are brought together.[25] Not only has the

---

[25]This Court in U.S. v. Adams, 383 US 39 (1966) said:

"It begs the question . . . to state merely that magnesium and cuprous chloride were individual known battery elements. If such combination is novel, the issue is whether bringing them together as taught by Adams was obvious in the light of prior art."

Petitioner refused to consider the arrangement of elements in Respondents' invention and how they cooperate with each other as directed by this Court, he has not identified any prior art which would render the Respondents' specific combination invention old. Judge Learned Hand in *Safety Car Heating & Lighting Co., Inc. v. General Electric Co., et al.*, 155 F.2d 937 (1946), eloquently stated the proper analysis as follows:

> "Substantially all patentable inventions are for combinations of old elements and what counts in determining whether a patentable invention has been made is the selection, out of all their possible permutations, of that new combination which will be serviceable."

If the claimed arrangement of elements and how they cooperate with each other is ignored by the Petitioner and no prior art is cited, it is clear that he could, with the same defective logic, hold as nonstatutory any combination invention ever invented and any invention still to be invented in the future.

This case can be distinguished over the *Flook* decision in still another way. In *Flook*, the Respondent was attempting to enlarge the legislative grant by claiming a process admittedly old by utilizing an unpatentable principle. He was attempting to ". . . transform an unpatentable principle into a patentable process. . . ."[26] In this case the reverse is true. The Petitioner is attempting to diminish the legislative grant by refusing to consider the Respondents' novel combination invention as statutory subject matter. The Petitioner refuses to properly examine the Respondents'

---

[26]*Parker v. Flook*, 98 S. Ct. 2526.

invention in his attempt to eliminate patent protection for computer inventions which he erroneously labels as a program or an algorithm. The Petitioner also relies on the old crutch of administrative convenience to justify his position of diminishing the scope of patent protection available under Title 35 of the United States Code. The reasoning of the Patent and Trademark Office is contrary to the legislative intent of Congress[27] and the decisions of this Court.[28]

It is ironic that at a time when the President of the United States has announced that he plans to "develop and maintain a climate conducive to industrial innovation" and "strengthen the patent system," the Petitioner is attempting to weaken it by alleging *inter alia,* that it would be more administratively efficient to deny a patent to the Respondents' complex invention without being required to search the prior art.[29] There is

---

[27]To promote the progress of Science and the Useful Arts . . .

[28]See Graham v. John Deere Co., 383 U.S. 1 (1966); McKay Radio & Telegraph Co. v. Radio Corporation of America, 306 U.S. 86 (1939); Eibel Process Co. v. Paper Co., 261 U.S. 45 (1923); and United States v. Adams, 383 U.S. 39, 50 (1966).

[29]The following are excerpts form the text of President Carter's message to Congress October 31, 1979 outlining his plan to revitalize America's industrial base.

> "TO THE CONGRESS OF THE UNITED STATES: Industrial innovation–the development and commerialization of new products and processes–is an essential element of a strong and growing American economy. . . . Further, industrial innovation is necessary if we are to solve some of the Nation's most pressing problems."
>
> "Patents can provide a vital incentive for innovation. . . . At my direction, the Patent and Trademark Office will undertake a major effort to upgrade and modernize its

(continued)

sufficient statutory and judicial law at present to correctly decide the patentability of any computer invention and the Patent and Trademark Office should be given a signal that it must follow this established law. The Court can do this by refusing to grant the Petition for Certiorari.

The decision of the United States Court of Customs and Patent Appeals below is another example of that court's clear understanding of computer inventions, the proper application of the statute 35 USC 101 to such inventions and the decisions of this Court relevant thereto. It is submitted that the United States Court of Customs and Patent Appeals has correctly applied the applicable decisions of this Court. When nonstatutory subject matter is involved, which the Appellants allege is statutory, the United States Court of Customs and Patent Appeals properly affirms the Patent and Trademark Office.[30] When on the other hand statutory

---

(footnote continued from preceding page)

process in order to restore the incentive to patents–and ultimately develop–inventions."

"We must also develop and maintain a climate conducive to industrial innovation. The Federal Government must take the lead in creating that climate."

From Congressional Quarterly Inc., November 10, 1979 (pages 2557-2558).

[30]In the Matter of the Application of Jean E. Maucorps (CCPA No. 79-554), decided 11/1/79 the court in affirming the Board of Appeals properly recognized that even though nonstatutory subject matter was "expressed" and claimed as an apparatus the invention was not patentable under 35 USC 101. The Court clearly seeing through the subterfuge said:

"That 35 USC 112 authorizes the claiming of 'mean for' performing a function cannot rescue Appellants' claim from

(continued)

subject matter is involved, which the Patent and Trademark Office alleges is nonstatutory, the Court of Customs and Patent appeals correctly applies 35 USC 101.[31]

In view of the fact that the *Flook* decision is not applicable to this case because, inter alia, no mathematical algorithm, principle, formula or idea is involved, there is no reason to even consider In re Diehr, 602 F.2d 982, (CCPA 1979) in connection with this case.[32] Whatever decision is made by this Court in its judgment regarding the Petition to be filed by the Commissioner of the Patent and Trademark Office in the Diehr case, it should be based solely on the merits of Diehr. The inventions in the two cases are so completely different that no benefit would inure to the Court in attempting to consolidate. On the contrary, in view of the complex nature of this invention, confusion would result from attempting to combine it with any other case.

(footnote continued from preceding page)

> the requirement of Paragraph 202, because Paragraph 221 does not authorize the claiming of apparatus entirely in terms of 'means for' performing a nonstatutory method."

Other cases where the United States Court of Customs and Patent Appeals affirmed the United States Patent and Trademark Office are as follows: In re Waldbaum, 559 F.2d 611, 194 USPQ 465 (1977); In re Richman, 563 F.2d 1026, 195 USPQ 340 (1977); In re de Castelet, 562 F.2d 1236, 195 USPQ 439 (1977); and In re Sarkar, 588 F.2d 1330, 200 USPQ 133 (1978).

[31]In re Johnson, Parrack and Lunsford, 589 F.2d 1070, 200 USPQ 199 (1978). In re Gelnovatch, 595 F.2d 31, 201 USPQ 136.

[32]See PW-13.

SUMMARY

The *Flook* and *Benson* decisions are not applicable to this case.

The Respondents' invention comprises a computer hardware machine implemented as a novel/hardware including a firmware element.[33] There is no question that the invention is a machine; even the Petitioner has admitted it is a machine. The applicable statute, 35 USC 101, explicitly states that new machines are statutory subject matter.

To present even a colorable case to this Court, the Petitioner has "labeled" the invention as a program. He has continually made this and other factual statements, which are totally speculative and which are not supported by the record in this case. Many of these statements are clearly erroneous. (See Appendix B). These speculative statements cannot be substantiated before this Court by the Petitioner without previous findings of fact. The Petitioner has also tried to distill the invention to one element—the firmware—contrary to this Court's decisions. The Petitioner also wrongly implies the firmware element would operate like a program in any computer machine.[34] The firmware

---

[33]Firmware is defined as "logic circuits in read only memory. . . ." See page 186 of Computer Dictionary and Handbook, by C. J. Sippl and C. P. Sippl. Howard W. Sams & Co. Inc., Indianapolis, Ind.

[34]The invention is marketed as an integral part of a new computer machine. On the other hand, computer programs are generally embodied on punched cards, magnetic tapes, or the like, and are licensed or sold as individual products, separate and apart from the computer machine.

element cannot run in any computer machine. It can operate only in combination with the other claimed elements.

A patent granted on this invention as claimed would not protect the firmware element per se. Any person could use the firmware element alone or as part of an unclaimed combination with other elements without infringing the patent.

The Petition to this Honorable Court for a Writ of Certiorari to the United States Court of Customs and Patent Appeals is based on a false premise—that this machine is a program. The Petitioner is obviously confusing a program with a machine that is programmable. The Petitioner's arguments are therefore totally without merit and based on a wrong assumption which inevitably educe wrong conclusions. A computer machine by any other name is still a computer machine.

CONCLUSION

There is no clear issue of law presented to this Court by the Petitioner. The Petitioner is asking this Court to legislate on a fact issue for which no factual record has been developed, and is not asking this Court to adjudicate an issue of law. For this and the reasons discussed above the Petition for a Writ for Certiorari should be denied.

Respectively submitted,

NICHOLAS PRASINOS
Attorney for Respondents
Honeywell Information Systems Inc.
Office of the General Counsel
200 Smith Street
Waltham, Massachusetts 02154
(617) 895-6129

Of Counsel:

F. F. DRISCOLL
H. L. HANSON
R. T. REILING

APPENDIX A

In re Application:	)	
	)	
John J. Bradley	)	Patent Appeal
and Benjamin S. Franklin	)	No. 79-533
	)	
Serial No. 570,331	)	

In the matter of the above referenced case which was heard before the United States Court of Custom and Patent Appeals, I was instructed to obtain the tape of the oral arguments from the court, listen to and transcribe the particular portion by the Solicitor.

I hereby certify that the following is a transcription of the portion requested:

> "Do you agree with Appellant that he is claiming a machine where it starts out, 'In multiprogram computer system . . .'?"

Answer:

> "I suppose it is a machine, it is an apparatus of some sort, yes Your Honor."

Date: December 7, 1979

Signed Carolyn L. Leonard
2043 Crystal Plaza Drive
Arlington, Virginia 22202

APPENDIX B

The following erroneous statements by the Petitioner are examples of the arbitrary and speculative pronouncements for which no factual record has been developed in this case, and illustrate the confusion and false conclusions of the Petitioner regarding this invention:

Erroneous statement of Petitioner:

> "Although the original PROMs could be programmed only once, many large computers now have a "read-write" PROM, which permits the user to reprogram the control memory any number of times electrically." (PW 4 footnotes).

Correct statement of fact:

There is no such thing as a read-write PROM. A PROM by its definition is a programmable read-only memory.

Erroneous statement of Petitioner:

> "Bradley's firmware is stored in a control memory of the PROM type which can be microprogrammed by the user." (PW 5 footnotes).

Correct statement of fact:

The term "user" as used in the Record means the computer manufacturer. The type of PROM used by Respondents require physical alterations by the manufacturer of the computer to create the desired circuits. Respondents' control memory cannot be reprogrammed by anyone.

Erroneous statement of Petitioner:

> ". . . the microprogram steps are disclosed in the flow charts of Figures 15b and 15c." (PW 5 footnotes).

Correct statement of fact:

Machine designers prepare a flow chart of their hardware design before building the machine. Figures 15b and 15c are flow diagrams of machine operations of one element of the Respondents' invention. Figures 15b and 15c do *not* illustrate microprogram ***steps***.

Erroneous statement of Petitioner:

> "Thus, he [the Examiner] concluded that the invention 'resides in an instruction in firmware which changes upon execution, the information in the system base.' " (PW 5 footnotes).

Correct statement of fact:

The claimed invention is a unique combination of hardware elements which cooperate and function with each other to achieve the objects of the invention on pages 5c,6c Appendix C.

Erroneous statement of Petitioner:

> "Any 'specific new, novel and unobvious arrangement of these structures . . . would appear to be provided by the microprogramming. . . ."

Correct statement of fact:

Microprogramming cannot arrange hardware structures as claimed--only the inventors can do that. The use of microprogramming is not an issue in this case.

Petitioner erroneously implies that:

> The invention is no different than a program for determining a "faster route between Washington and New York." (PW 12).

Correct statement of fact:

The Respondents' invention is a new machine not a program. A complete reading of the specification makes this clear. (Appendix C, pps. 1c-109c). The cities of Washington and New York are known and all paths between them whether or not already discovered are in existence. Merely discovering the shortest path whether with the aid of a computer program or otherwise does not breathe "newness" to the discovery of an existing path. Such a hypothetical discovery would not be statutory subject matter under 35 USC 101 in view of this Court's reasoning in *Flook*. The Respondents' invention, however, is a new combination of elements *which combination was not known prior to its creation by the Respondents.*

APPENDIX C

APPLICATION OF JOHN J. BRADLEY AND BENJAMIN S. FRANKLIN, FILED APRIL 21, 1975, SERIAL NUMBER 570,331, FOR SWITCH SYSTEM BASE MECHANISM.

ABSTRACT OF THE DISCLOSURE

A model-independent instruction in firmware is provided to change, upon execution, information in the system base which is located in both main memory and internal scratch pad registers.

RELATED APPLICATIONS

The following applications are incorporated by reference to the instant application.

1. "Buffer Store" invented by J.L. Curley, T.J. Donahue, W.A. Martland, and B.S. Franklin, filed on October 5, 1972 having Serial Number 295,301 and assigned to the same assignee named herein.
2. "Variable Masking for Segmented Memory" invented by Wallace A. Martland and John L. Curley, filed on October 5, 1972 having Serial Number 295,303 and assigned to the same assignee named herein.
3. "Override Hardware for Main Store Sequencer" invented by Thomas J. Donahue, filed on October 5, 1972 having Serial Number 295,418 and assigned to the same assignee named herein.
4. "Main Memory Sequencer" invented by T.J. Donahue, J.L. Curley, B.S. Franklin, W.A. Martland, and L.V. Cornaro, filed on October 5, 1972 having Serial Number 295,331 and assigned to the same assignee named herein.
5. "Main Memory Reconfiguration" invented by J.L. Curley, B.S. Franklin, W.A. Martland, T.J. Donahue and L.V. Cornaro filed on October 5, 1972 having Serial Number 295,417 and assigned to the same assignee named herein.
6. "Ring Checking Hardware" invented by _____ filed on the same date as the instant invention, having Serial Number _____ and assigned to the same assignee named herein.
7. "Segmented Address Development" invented by _____, filed on _____ and having Serial Number _____ and assigned to the same assignee named herein.
8. "Protection of Information in a Multiprogram Multiprocessor Computer System" invented by _____, filed

on _____ and having Serial No. _____ and assigned to the same assignee named herein.

9. "Ring Crossing Mechanism" invented by _____, filed on _____, and having Serial Number _____ and assigned to the same assignee named herein.

10. "Procedure Calls and Stack Operations", invented by _____, filed on _____, and having Serial Number _____ and assigned to the same assignee named herein.

11. "Process Synchronization Utilizing Semaphores" invented by _____, filed on _____, and having Serial Number _____.

12. "Arrangement for Automatically Managing, Scheduling and Allocating System Resources" invented by _____, filed on _____ and having Serial Number _____ and assigned to the same assignee named herein.

13. "Data Processing Memory Management System" invented by _____, filed on _____ and having Serial Number _____ and assigned to the same assignee named herein.

14. "Arrangement for Initializing a Computer System Controlled by Reloadable Site-Dependent Information Structures", invented by _____, filed on _____ and having Serial Number _____ and assigned to the same assignee named herein.

15. "System Base", invented by _____, filed on _____, and having Serial Number _____ and assigned to the same assignee named herein.

16. "Absolutization During Initialization", invented by _____, filed on _____, and having Serial Number _____ and assigned to the same assignee named herein.

BACKGROUND

Field of the Invention:

This invention relates generally to computer systems and more particularly to a system and method for communicating between hardware and software in a multiprogramming/ multiprocessing environment.

Multiprogramming/multiprocessing computer systems require the use of a communications area in main storage for communications between hardware and software. (See the application entitled "System Base" invented by _____ filed on _____ having Serial Number _____ and assigned to the instant assignee, and which application has been incorporated by reference herein). The system base is generally located at a fixed location in main memory, although this location may be changed or relocated through the use of a device such as a boundary address register (BAR). (See invented by _____, filed on _____, having Serial Number _____ and assigned to the instant assignee, and further incorporated by reference in the instant application).

High performance computing systems find it necessary to store portions of the system base in internal scratchpad registers instead of main memory, and hence inaccessible and invisible to software running in the system. Accordingly, a problem results when software (i.e. the programmer) wants to change positions or change information of the system base which happens to be located in these internal central processor scratchpad registers which software cannot access.

Prior Art:

The problem has been solved in one of two ways. The first method is to simply re-initialize the entire system, loading the system base with different contents as a result of a new initialization process. This technique is extremely time-consuming and quite undesirable.

A second technique for solving the problem is to have the software employ special model-dependent properties of the particular machine on which it was running, to change the system base. This results in the very undesirable property of completely model-dependent software—a feature which certainly does not serve to endear a computer manufacturer to a customer.

What is needed therefore is a technique of changing information in the system base both in main memory and internal scratchpad registers making the system base in scratchpad memory accessible to software but not visible to software i.e. cannot be read by the software.

OBJECTS OF THE INVENTION

It is a primary object of the invention to provide an improved system and method for changing information in the system base of a computer system.

It is another object of the invention to provide a switch-system base instruction in firmware which changes, upon execution, information in the system base.

It is still another object of the invention to provide a system and method for changing information in the system base which is located in main memory and scratchpad registers.

Yet another object of the invention is to provide an improved switch system base mechanism.

These and other objects of the invention will become apparent from the description of a preferred embodiment of the invention when read in conjunction with the drawings contained herewith.

SUMMARY OF THE INVENTION

A model-independent instruction in firmware is provided to change, upon its execution, information in the system base which is located in both main memory and internal scratch-pad registers.

The system base located at a location specified by the boundary address register BAR in main memory, is copied into a new location in main memory with predetermined portions of its information changed. This is readily accomplished by the operating system. (See application incorporated by reference into the application herein). The switch system base instruction then transfers the information which is in the system base at the new location into the system base at the location specified by BAR and also that portion of the system base located in scratchpad memory.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features which are characteristic of the invention are set forth with particularity in the appended claims. The invention itself, however, both as to organization and operation together with further objects and advantages thereof may best be understood by references to the following description taken in conjunction with the drawings in which:

Figure 1 is a block diagram of a multiprogramming system utilizing the invention.

Figure 2 is a schematic representation of various hardware structures utilized by the invention.

Figure 3 is a legend of terms used for reserved areas of storage in registers depicted in Figure 2.

Figure 4 is a schematic diagram of a process control block.

Figure 5 is a schematic diagram of a system for addressing a process control block.

Figure 6 is a schematic diagram of the system base of the invention.

Figure 7A and 7B are a schematic representation of a stack segment and a stack frame respectively.

Figure 8 is a schematic diagram of a system for addressing G-segments and in particular the queue of processes in the GO segment.

Figure 9 is an exploded schematic diagram of a GO segment illustrating queue of processes and process linking.

Figures 10a through 10L are block diagrams of structures in the PCB.

Figures 11a through 11r are block diagrams of structures in the system base.

Figure 12 is a schematic diagram of addressing schemes of user and system segments utilizing the system base and PCB structures.

Figure 13 is a schematic diagram of the control unit.

Figures 14a through 14i are flow diagrams of the dispatcher unit in firmware.

Figure 15a is a structure of the switch system base instruction.

Figures 15b-15c are flow diagrams of the switch system base instruction in firmware.

DESCRIPTION OF A PREFERRED EMBODIMENT

GENERAL DISCUSSION

The invention operates typically in the hardware system environment hereinafter described coordinated by a hardware/firmware/software operating system. Referring to Figure 1 the subsystems are the processor subsystem 101, the storage subsystem 102, and one or more — up to 32 — peripheral subsystems 103. The processor subsystem contains a central processing unit (CPU) 104 and up to four input/output control units (IOC) 105. Each peripheral subsystem consists of a peripheral control unit (PCU) 106, a number of device adapters (DA) 107, and up to 256 peripheral i/o devices 108. The storage subsystem 102 consists of one to four semiconductor memory modules of 32 to 512 kilobytes each.

I. PROCESSOR SUBSYSTEM

In the processor subsystem 101, the CPU 104 performs the basic processing operations for the system, and interfaces with memory 102. The IOC 105 controls all information exchanges between the storage subsystem 102 and peripheral devices 106.

A. CENTRAL PROCESSING UNIT

The CPU includes a main memory synchronizer 109, a buffer store 110, various elements that comprise the computational unit 111, and optional emulation facilities 112. The main memory synchronizer 109 resolves conflicts for the use of main memory among the computational unit

111, the buffer store 110, and the IOC 109. Conflicts are resolved on a priority basis: the IOC has the highest priority followed by memory writes (from the computational unit) and then memory reads (into the buffer store). The main CPU also includes the address control unit ACU 131 which controls main memory addressing and the associative memory AS 132 used to store most recently used addresses of main memory. The buffer store 110 is a small high-speed buffer memory that reproduces a selected region of main memory and interfaces with the computational unit to decrease average memory access time. During each memory read, both the buffer store and main memory are accessed. If the information to be fetched is already in the buffer store, the main memory read is terminated and the information fetched from the buffer store. Otherwise the main memory 102 is read. Every time this is done, the CPU 104 fetches 32 bytes that contains the desired information. This information remains in the buffer store for future memory references. Since the buffer store is transparent to software, the program controlling the computer at any given moment cannot determine whether the information it is processing has been fetched from the buffer store or from main memory.

The computational unit 111 performs all data processing and address generation within the CPU. A typical control store 130 within the computational unit (see a book entitled Microprogramming: Principles and Practices, Samir S. Husson, Prentice Hall, Inc.) contains firmware which initializes the system, controls the CPU 104 and IOC 105, and decodes an instruction set (not shown). Optionally the control store may provide scientific instructions, test routines, emulation packages, or special purpose features which extend the capabilities of the processor subsystem.

As an option, the CPU provides emulation of systems other than the instant system. Emulators 112 are com-

ponents of firmware, software, and in some instances hardware.

B. INPUT-OUTPUT CONTROL UNIT

The IOC 105 portion of the processor subsystem provides a data path between any peripheral subsystem 103 and the storage subsystem 102. This path allows for the initiation of peripheral commands and controls the resulting data transfers. An IOC can typically handle up to 32 channel control units (not shown).

C. PERIPHERAL SUBSYSTEMS

In a peripheral subsystem 103 on Figure 1 the PCU 106 is a stand-alone microprogramming processor that relieves the load on the CPU 104 by controlling the i/o devices 108 during i/o operations. The PCU does this by executing instructions contained in a channel program. This program results in arithmetic, logical, transfer, shift, and branch operations being performed in the PCU. There are several kinds of PCU's according to the kind of device each controls: i.e. unit record, mass (disk) storage, magnetic tape, communications, etc.

Device adapters 107 mediate between every PCU and the devices it controls. Each contains the dedicated firmware and logic necessary to implement communication with a particular type of device. Depending on the type, a DA 107 controls one or several devices.

The major functions performed by a peripheral subsystem 103 are as follows:

1. Transforming CPU instructions into a series of commands acceptable to the appropriate peripheral device.
2. Packing and unpacking data in the form needed by the CPU or the appropriate peripheral device.

3. Keeping the CPU informed of the status of the subsystem and of the devices under its control.
4. Independently initiating and processing error and recovery procedures.
5. Allowing on-line diagnosis of a device without disturbing the device-sharing capabilities of the associated peripheral processor.

A PCU resolves conflicts for main memory between devices attached to it; however, the IOC resolves conflicts between PCU's.

D. STORAGE SUBSYSTEM

Each memory module 1-4 is 4 or 8 bytes wide. The number of modules, their size, and the data path width may vary according to size of computer. Memory modules are four-way interleaved in such a way that the four modules are accessed sequentially (module 1 contains the first 8 bytes, module 2 contains the second 8 bytes, etc.). Interleaving decreases the number of conflicts for access to main memory and thereby decreases the average memory access time. Memory is reconfigurable in case of failure; i.e., blocks of memory within a module may be removed without destroying contiguous addressing.

Main memory 102 consists of a capacitive storage medium in the form of metal oxide semiconductor (MOS) chips. This medium operates on the refresh principle to maintain information. Each memory location is typically refreshed at least once every 2 milliseconds; the design ensures that few conflicts occur between refresh timing and memory accesses. (In cases of conflict, refreshing takes precedence).

An area at the beginning of main memory is reserved for hardware and firmware. The upper limit of this area is

defined by the content of a boundary address register (BAR - to be later described) which is visible to the system software. The BAR content is set at system initialization time. The memory area below the address specified in the BAR can contain IOC tables which define the configuration of the peripheral subsystems, firmware to control the CPU, or microprograms and tables for emulation. The size of the area below the address specified in the BAR depends on the system configuration. Whether microprograms are in main memory or control store depends on the system configuration and the applications run on the system.

II. BASIC MACHINE STRUCTURES

There are typically three basic data structures utilized in this hardware: data formats, software visible registers, and the instruction formats.

A. DATA FORMATS

Information is transferred between memory and the CPU in multiples of 8 parallel bits. Each 8-bit unit of information is called a byte. Parity or error correction data is also transferred with data but cannot be affected by software. Therefore, in this patent specification the term data excludes the associated parity or error correction data.

B. BYTES

Bits within a byte are numbered 0 through 7 from left to right. Bytes are processed separately or in groups. Two bytes constitute a halfword, 4 bytes a word, 8 bytes a doubleword, and 16 bytes a quadword. These are the basic formats for all data, including instructions.

C. DATA REPRESENTATION

All data are in binary form, but may be interpreted as binary, decimal, or alphanumeric. Data bits are interpreted in groups of four, as binary coded decimal data; eight as alphanumeric, or 16 to 64 as binary digits. The latter are interpreted as signed, fixed, or floating-point numbers in binary notation. Any number of contiguous bits up to a doubleword may also be manipulated as a string. The alphanumeric character set is represented in EBCDIC. ASCII is supported as an alternate exchange code.

D. BYTE ADDRESSES

Byte locations in main memory are consecutively numbered starting with zero; each number is the address of the byte. A group of consecutive bytes is said to be halfword-, word-, doubleword-, or quadword-aligned, if the address of the left byte in a group is a multiple of 2, 4, 8, or 16, respectively. Whenever a halfword, word, doubleword, or quadword is so aligned, that unit can be fetched from that address. The location of data in main memory is specified by a data descriptor which is accessed indirectly during address development. (See Patent Application No. _____ filed _____ entitled Segmented Address Development and assigned to the same assignee as the instant application).

E. VISIBLE REGISTERS

There are 33 user-visible registers in the CPU 104 Figure 1 whose contents collectively define the state of the CPU. There are four types: (See Figure 2).

1. general registers

2. base registers
3. scientific registers (optional)
4. miscellaneous registers

F. GENERAL REGISTERS

General registers (GR) 201 are used to manipulate fixed-point binary numbers and bit strings. There are typically sixteen 32-bit general registers in the CPU 104—GR0 through GR15. General register GR8 through GR15 are also usable as index registers. When used as index registers, they are herein called X0 through X7: Indexing is performed using the 32-bit two's complement integer contained in a register.

G. BASE REGISTERS

Base registers (BR) have the same format as instruction counters IC and stack registers 202-203. Base registers are used during address computation to define a part of memory. There are typically eight 32-bit base registers, BR0 through BR7.

H. SCIENTIFIC REGISTERS

Scientific registers (SR) are optional equipment for computation with floating-point binary numbers. There are typically four 8-byte scientific registers which are referred to as SR0 through SR3. Scientific registers have the format 204-205 of Figure 2.

I. MISCELLANEOUS REGISTERS

There are five other registers:

- instruction counter - having format 202-203;

- status register - having format 207;
- stack register (called the T register);
- boundary address register - having format 202-203; and
- hardware control mask register - having format 208.

The instruction counter (IC) is a 32-bit register that contains the address of the instruction being executed. The status register (STR) 207 is an 8-bit register that records facts about the procedure currently being executed, for example, whether an underflow was caused by the most recent operation. The stack register also known as the T-register is a 32-bit register that contains a pointer to the top of a pushdown stack associated with the currently active procedure. Stacks to be described infra provide a work space, and a mechanism for saving local variables and preserving procedure entry, and return information. The boundary address register (BAR) 206 is a 28-bit register which specifies the lowest absolute main memory address accessible by software. This register is loaded during system initialization and can only be read by software. The hardware control mask register 208 is an 8-bit register which records machine condition information.

J. INSTRUCTION FORMATS

There are approximately 200 instructions although more or less may be utilized. Each instruction is one of four different lengths but always an even number of bytes long. Instructions are stored in consecutive storage locations. The address of the leftmost byte is a multiple of 2, and is the address of the instruction.

The eight most significant bits (and in some cases bits 8 through 11 or 12 through 15) of an instruction represent the operation code, while the remaining bits represent one or

more operands. An operand may be a register designator, displacement designator, address syllable (logical address) literal value, immediate literal value. The type and number of operands are determined by the instruction format.

III. SYSTEM ORGANIZATION

A. JOB STEP AND TASK

Work to be performed by the computer system is defined externally by a series of job steps via a job control language. A job step is a unit of work to which hardware resources are allocated. Typically a job step consists of several tasks. A task is the smallest unit of user defined work consisting of a stream of instructions executed without parallelism.

B. PROCESS

The user-visible concepts of task and job step are represented in the hardware by a process and process group, respectively. A process is defined as an ordered sequence of instructions which can be executed asynchronously by the CPU (i.e., several processes can be active and sharing resources, but only one process is actually running at any one instant). A process group is a related set of processes necessary to perform one job step.

C. PROCESS CONTROL BLOCK AND SYSTEM BASE

Because processes can relinquish CPU control at various points during their execution, a storage area in main memory is made available to a process to save CPU status. This status information is utilized to precondition the CPU before a process regains control of the CPU.

The storage area assigned to a process is called a process control block (PCB) 400 on Figure 4. The data contained in a PCB include the addresses of memory areas (address space) assigned to the process, the contents of all pertinent registers, and the state of the process. Thus a PCB serves as a temporary storage area for information necessary to start or restart a process without any information loss. Each PCB is visible to the hardware and can be addressed by the operating system via a set of hardware tables developed during system initialization and modified during system operation (Figure 5).

There is an absolute main memory area which is referred to as the system base (Figures 5 and 6). This area is developed by firmware and is accessible via the base address register (BAR) 501 which can be read but not written. The system base 502 contains a number of system attributes which include a job step number and a process group number (J, P) respectively for the currently running process. Another attribute in the system base is a pointer to a hardware defined data structure known as the J table 503. This table contains an entry for every job step presently in the system. Each entry in the J table 503 points to an associated P table 504 which is also a hardware defined data structure. This table defines a process group and contains an entry for every process in the process group. Each P-table entry points to a PCB 400.

Referring to Figure 5 the J-table pointer 505 indexed by the J number via computational unit 506 provides access to a J-table entry 503. This entry contains a P-table pointer which when indexed by the P number via computational unit 506 provides access to a P-table entry 504. The P-table entry contains a pointer 507 to the PCB of the current running process. Thus the operating system can access the active PCB using the contents of the BAR 501 and can access any other PCB given its associated (J, P) logic name.

D. MEMORY SEGMENTATION

In a multiprocess environment, such as herein described there are many processes in memory at any given time. These processes vary in size and demand for memory which causes a memory allocated problem. The hardware herein described in cooperation with an operating system (not shown herein) solves the problem by dynamically allocating memory space. Due to the random nature of memory requirements, memory is allocated in variable size segments and the memory allocation can be restructured during process run time. Thus, a process may be allocated a number of noncontiguous memory segments. This memory allocation method is called segmentation.

Segmentation presents an additional problem in that memory addresses have to be modified whenever part or all of a process is relocated. To alleviate this problem the system herein described provides a technique whereby addresses used by a process are logical rather than absolute main memory addresses. These logical addresses are used to develop absolute addresses.

Segmentation also allows each process to access its own or related memory segments via a system of segment descriptors. By accessing a segment descriptor, a process can obtain the address of a segment. Segment descriptors are contained in main memory and are maintained by the operating system.

Each process may have access up to 2068 memory segments. Normally, this would require an equal number of segment descriptors per process. However, since segments can be shared, the operating system groups segment descriptors into segment tables. This grouping is based on accessability by one process (task), a process group (job step), or globally (system wide). Each process may have up

to 15 segment tables associated with it. This technique requires only one segment descriptor for each segment which can be accessed by a process via a segment table. Thus, the memory space required for segment descriptors is decreased; memory updating during relocation is reduced; and some program protection is provided. (The main mechanism for program protection is the ring system. See Patent Application No. _____ filed _____ entitled "Protection of Information in a Multiprogram, Multiprocessor Computer System;" also see Patent Application No. _____ filed _____ entitled "Ring Crossing Mechanism" both assigned to the same assignee as the instant invention.

A process must be able to determine which segments it is allowed to access. Accordingly, the system provides a process with two segment table word arrays (STWA). These arrays contain the addresses of all segment tables accessible to a process. There are two segment table word arrays per process because there are two segment sizes, large and small. Large segments have a maximum size of 2^{22} bytes while small segments have a maximum size of 2^{16} bytes. All segments vary in size in 16-byte increments up to the maximum. A system can typically accomodate up to 28 large segments and 2040 small segments.

Segment table word arrays may be relocated by the operating system; therefore, a process must know the absolute address of its associated STWA's. The PCB for any process contains two words which contain this information which are known as address space words ASW0-1 on Figure 4. Each word points to a segment table word array STWA. The operating system updates the contents of the ASW's whenever the associated STWA's are relocated. Working down the chain of pointers and decoding the segment descriptor is a firmware function and thus once initiated is not visible even to the operating system.

Segmentation defines over 200 million bytes of address space as being available for processes. This number exceeds the capacity of main memory; therefore, a secondary storage (magnetic disk or drum) is used in conjunction with main memory. The operating system creates the illusion that the system has a much larger main memory than is really available. This concept is called virtual memory.

At any given time, a defined segment may or may not be physically in main memory. The contents of a segment descriptor indicates whether or not the associated segment is in main memory. The hardware detects any attempts by a process to access a segment not in main memory and notifies the operating system. The operating system causes the desired segment to be loaded into main memory from secondary storage. Then the operating system places the segment's memory address in the segment descriptor which is the only place where the absolute address of a segment can be found. This operation is invisible to the process and thus it is not aware that the segment was not in main memory or that it may have to be relocated in main memory. (For details on memory segmentation see Patent Application No. _____ filed _____ and entitled "Segmented Address Development". Also see patent No. _____ .)

The computer system herein described provides data and procedure protection by preventing processes from interferring with each other or sharing each other's address space in an unauthorized manner. This protection is accomplished by restricting addressability via memory segmentation and by a ring system.

The segment tables isolate the address space of the various processes in the system. Processes always use a segmented address during execution. A segmented address consists of a segment number and a relative address within the segment (see above referenced application on Segmented Address

Deve' ›pment). The hardware checks that the address used by a process is part of the address space assigned to the process. If the address is outside the prescribed address space, an exception occurs. A process cannot refer to data within the address space of another process because the hardware uses the segment tables of the referencing process. Thus, there is no possibility for a process or process group to reference an entity belonging to another process group.

Generally, overlap in address space in the system occurs for those segments shared by all processes. These public segments are created by system programs which check to insure against address conflicts. Thus, segmentation protects user programs against each other and protects the operating system against user programs.

Segments shared by several processes are not protected from misuse by one of these processes. To solve this problem, a ring system is utilized whereby procedure and data segments are grouped into a four-class hierarchy. The four ring classes are numbered 0 through 3. Each ring represents a level of system privilege with level 0 (the innermost ring) having the most privilege and level 3 (the outermost ring) the least. Every procedure in the system has a minimum and a maximum execute ring number assigned to it which specifies who may call the procedure. A procedure is a subroutine which is capable of calling other procedures and passing parameters to them:

The general rules of the ring system are as follows:

1. A procedure in an inner ring has free access to data in an outer ring. Conversely a procedure in an outer ring cannot access data in an inner ring.
2. A procedure in an outer ring can branch to a procedure in an inner ring, but the reverse is not allowed.
3. Each segment containing data is assigned two ring values, one for read (RD) and one for write (WR).

These ring values specify the maximum ring value in which a procedure may execute when accessing the data in either the read or write mode.

Each time a procedure instruction is executed, the procedure's ring number (effective address ring, EAR) is checked against the ring numbers assigned to the segment containing the referenced data. The EAR is the maximum number of process ring numbers in the instruction counter and all ring numbers in base registers and data descriptors found in the addressing path. Access to the data will be granted or denied based on a comparison of the ring numbers. For example, if a system table exists in a segment having a maximum read ring value of 3 and a maximum write ring of 1, then a user procedure executing in ring 3 may read the table but may not update the table.

By predesign, rings 0 and 1 are reserved for the operating system and rings 2 and 3 are reserved for the user. Ring 0 contains those segments critical to total system operation. Ring 1 contains the bulk of the system segments whose failure would not be catastrophic and would allow recovery. The user may utilize ring 2 for checked-out programs and ring 3 for programs being debugged.

F. PROCEDURE CALLS

The procedure call is an important function in the system herein described. Procedure calls are used to pass from one procedure to another; to allow user procedures to employ operating system services; and to achieve a modular structure within the operating system. A procedure call is effected by instructions and a hardware recognized entity called a stack (Figure 7A).

A stack is a mechanism that accepts, stores and allows retrieval of data on a last-in-first-out basis. Stacks reside in

special segments called stack segments. A stack segment consists of a number of contiguous parts called stack frames 701 (Figures 7A and 7B) which are dynamically allocated to each procedure. The first stack frame is loaded into the top of the segment and succeeding frames are loaded after it. The last frame loaded is considered the top of the stack. The T-register 702 locates the top of the stack for the currently active process. A virtual T-register exists in the PCB of all other processes in the system.

A stack frame 701 of Figure 7B consists of three areas: a work area 702 in which to store variables, a save area 703 in which to save the contents of registers, and a communications area 704 in which to pass parameters between procedures. Prior to a procedure call, the user must specify those registers he wishes saved and he must load into the communications area the parameters to be passed to the called procedure. When the call is made, the hardware saves the contents of the instruction counter IC and specified base registers to facilitate a return from the called procedure.

Each procedure call creates a stack frame within a stack segment 701 and subsequent nested calls create additional frames. Each exit from one of these called procedures causes a stack frame to be deleted from the stack. Thus, a history of calls is maintained which facilitates orderly returns.

To insure protection between procedures executing in different rings, different stack segments are used. There is one stack segment corresponding to each protection ring per process. A PCB contains three stack base words which point to the start of the stack segments for rings 0, 1 and 2 associated with the process. The ring 3 stack segment can never be entered by an inward call; therefore its stack starting address is not required in the PCB.

IV. PROCESS MANAGEMENT AND SYNCHRONIZATION

The system herein provides for multiprocessing operations which are controlled by an operating system using a combination of software, hardware and firmware. Software creates and deletes processes within the system while hardware and firmware multiplex processes on the CPU. In addition, a combination of software, hardware and firmware provide for synchronization between processes.

Processes are normally, but not always, started and stopped at the initiation and termination of i/o operations, during related job handling, and at other times for purposes deemed necessary by the operating system. Therefore, a communications system is necessary to efficiently start and stop related processes and to pass information between them. The hardware system herein provides internal messages called semaphores to provide a communications link between the processes.

A. PROCESS STATES

A process can be in one of four possible states at any time: running, ready, waiting or suspended. The hardware recognizes these four possible process states and executes various firmware procedures to effect process dispatching, state changes and to maintain data structures based on a process's state. The PCB contains a state field which defines the current state of its associated process.

A process is in the running state when it has control of the CPU. This state involves supplying the CPU with an address space (segment tables) and a starting address. The CPU then executes instructions in the procedure segments of the process. The process name J table word (logical

address) of the PCB for the currently running process is retained in the running process word (BAR +60) within the system base (Figure 6). (Note: The system base shown in Fig. 5 is the same as that shown in Fig. 6, but with some details omitted.)

The ready state is equivalent to running state except that the process does not have control of the CPU because it has not been recognized by the CPU. A process in the ready state is in contention for the CPU with other ready processes and the running process.

A process is in the wait state when it cannot continue until a specific event occurs such as a message via a semaphore. A waiting process is not in contention for the CPU but it may be in contention with other waiting processes for the required event.

A suspended process is a process which has been stopped for a time by software and may be resumed later. The decision to stop and resume the process is external to the process. Thus, a suspended process is not active and therefore cannot receive notification of event occurrences and cannot utilize the CPU.

A process is suspended under the following conditions:

(1) By executing a Terminate instruction (as a result of having completed all its functions.)

(2) By execution of a Suspend instruction by the operating system.

(3) By the occurrence of an exception condition whereby control is transferred to the operating system.

B. PROCESS DISPATCHING

Processes move from one state to another voluntarily by action of the process while running or involuntarily by the actions of other processes. CPU firmware, known as the

dispatcher, controls the transaction of processes between states. The dispatcher uses a set of queues (to be later described) to manipulate processes which are in the ready or the waiting states. Suspended processes are controlled by software.

Referring to Figures 6, 8 and 9, a ready or waiting process is represented by a PCB and a special queue entry called a process link. Figure 9 shows an exploded view of contents of the GO segment 802, and contains process links 803a-803b and 803c-803g of active processes, and free process links 805a-805c of suspended processes. Each process link specifies the process name (J, P), the process priority and a pointer to the next process link in the queue. There are various types of queues such as wait queue 803a-b and ready queue 803c-g.

A hardware device similar to the J table, known as the G table, (Figures 6 and 8) contains pointers to all general (known system wide) segments 802-802n. The first element, GO, of the G table 801 points to that segment 802 containing the dispatcher queues. A G-table pointer to the G table 801 is found in the system base 502 on Figure 5. Also in the system base is an entry called the internal process queue word (IPQW) which identifies the head 805 of the ready queue 803c-803g in the GO segment 802.

Thus, the dispatcher can examine all ready processes by consulting the ready queue 803c-803g. When the currently running process changes states, the dispatcher removes the process link at the head of the ready queue and uses the J, P name to access its PCB. The process defined by the PCB then becomes the new running process.

Since more than one process may be awaiting on the same event, a queue of waiting processes 803a-803b exists for each event. Waiting processes are also strung together via process links 805 residing in the GO segment. A pointer

to the head of a wait queue exists in a semaphore 903 (to be later described). A number of events exist for which a process may wait; therefore, there are a number of wait queues each of which has an associated semaphore 903, 904.

The number of processes ready or waiting varies dynamically. Thus, the number of process links required for the ready and wait queues also varies. This fact introduces a memory management problem for the dispatcher. The problem is solved by another queue called the free process link queue 805a-c. This queue links together all process links in segment GO that are not being used by the ready or the wait queues and can be used to extend a particular queue of ready or waiting processes. A pointer 901 to the head 902 of the free process link queue 805 resides near the beginning of the GO segment 802.

C. PROCESS SYNCHRONIZATION

Process synchronization is required to coordinate the activities of two processes working on the same task. The synchronization is achieved using semaphores 903-904 which are data structures residing in the address space of communicating processes. A semaphore is used to signal event occurrence and to handle queues of messages. An event in this context is anything observed by a process which may be of interest to some other process. The event may be the completion of an asynchronous operation or the availability of a resource.

A process uses two semaphore operations to signal an event occurrrence. One operation sends a signal to a semaphore; the other picks up a signal from a semaphore. (The sending operation is often called a V-operation; the receiving operation is called a P-operation). The sending

operation allows a process to send data or a signal that data are ready. The semaphore stores the signal until another process is ready to pick it up. Thus, the sending process is free to proceed, since it has sent the data. The receiving operation examines a specified semaphore and picks up the signal. If a signal is present, the receiving process continues executing. However, if there is no signal at the semaphore, the receiving process enters the wait state. The semaphore then serves as a pointer to the head of a wait queue. The process remains in the wait state queued at the semaphore until another process sends a signal to that particular semaphore. Thus, a semaphore can hold a signal until a process picks it up, or a semaphore can hold a process until a signal is sent to it.

Messages can also be passed from process to process. A message has the same present or not present quality as a signal plus additional information. Part of the information is supplied by hardware and part is supplied by the procedure of the process that sent the message. A message carries the process name of the sending process. Thus, many processes can send information through a single semaphore stamped with the sender's name.

A message semaphore may have a queue of messages waiting to be picked up by processes. As with signal semaphores, requirements for memory space increases and decreases thus presenting a memory management problem. Again, the problem is solved with a queue of free message links. These links reside in a known place in a segment that can easily be found when needed to supply or absorb message links.

Because semaphores and the queues built on them are shared by different processes, the total semaphore structure is protected. This is accomplished by hardware and software conventions that restrict access to any segment containing

semaphores. Thus, semaphores must be in semaphore descriptor segments, some of which may be G segments (if system communications is necessary). However, all G segments (except GO) are semaphore descriptor segments.

Each semaphore descriptor contains a pointer to a semaphore. Semaphore addresses are developed via a semaphore descriptor, thus providing added protection for the semaphore. A semaphore segment can be addressed logically using a segment number and a relative location within the segment or directly using the G, D number.

PROCESS CONTROL BLOCK STRUCTURES

Referring to Figure 4 there is shown the format of the process control block (PCB). The process control block 400 is a storage area in main memory made available to a process to save the CPU status. Addressing a PCB is performed as described supra in relation with Figure 5. The PCB pointer 507 (Figure 5) points to the process control block PCB at memory location 0 on Figure 4. It will be noted that proceeding in a downward direction memory locations increase by 4 bytes whereas in proceeding in an upward direction from memory location 0 they increase by 8 bytes. The downward memory locations are considered positive from 0 whereas the locations in an upward direction from 0 are considered negative directions. The upward locations are optional and may or may not be included in the process control block; also locations 148 through 176 are also optional. (Note that the numerals under memory location specify the displacement in bytes from the 0 reference location of the process control block PCB and are not to be confused with the reference numerals commonly used to identify parts in a patent drawing). Starting at byte 0 up to but not including byte 16 there are stored four process main

words PMW 0 through PMW 3 with each process main word PMW being four bytes in length. Process main word 0 occupies bytes 0 through 3 and is comprised of 4 parts: a capability byte, a priority byte, a state byte and a decor extension byte DEXT. Referring to Figures 10a through 10d there are shown details of process main word PMW 0, with further details of the capability byte 1001 shown on Figure 10b. Referring to Figure 10b, the first bit 1005 is the accounting mode bit for indicating whether or not time accounting functions are being performed for the process. When the accounting mode bit 1005 is set to binary 0 no time accounting function is being performed for the process; whereas when the accounting mode 1005 is set to binary 1, time accounting is being performed. The scientific mode bit 1006, when set to zero, indicates that saving of scientific register of the machine is not performed and the scientific register saving area located at bytes 148 to 176 on Figure 4 does not exist in the process control block PCB. When the scientific mode bit 1006, is set to binary 1, the scientific optional feature exists and is being used in the process, and the scientific registers saving area is used to save the contents of the scientific registers when necessary. The code mode bit 1007 indicates whether or not a standard code set or compatibility code set is being used by the process, with a binary 0 in that position indicating that standard code set is being used; whereas a binary 1 in the third bit position 1007 indicates a compatibility code set is being used. The remaining of the bits of the capability byte are set to zero.

Details of the priority byte 1002 are shown on Figure 10c. Referring to Figure 10c the first four bits 1008 of priority byte 1002 is utilized to set the priority level of the process associated with that given process control block PCB. Each process is assigned one of 16 levels of priority which is used for ordering competing processes i.e. (a) for choosing the

process to be run among ready processes, (b) for putting processes in queues. Priorities decrease from 0 to 15, and for a given priority level the FIFO (first in first out) rule is applied. The next 4 bits 1009 of priority byte 1002 are zeroes.

Referring to Figure 10d details of the state byte 1003 are shown. A state byte is utilized to provide information with regard to the process associated with the process control block PCB 400. The active field bit A 1010 is set to binary 1 when the process is activated. The suspend field S 1011 is set to binary 1 when the process is suspended. The substate field SS 1012 is a 2 bit field and defines the following substates of the process: (a) when set to binary 00 the process is inactive; (b) when set to binary 01 the process is waiting in the queue of ready process (Q/PR/RDY); (c) when set to binary 10 the process is waiting on a semaphore in a queue of semaphores (Q/PR/S); (d) when set to binary 11 the process is being executed by the processor. The mid-operation field (MOI) 1013 is set to binary 1 when an interrupt happens and is taken care of during the execution of an instruction—i.e. before the completion of the process. The extended decor mode bit EXTD 1014 is set to 1 when the process is operated in an extended decor mode which is an emulation mode of the machine. Bits 1015 and 1016 are set to 0. The fourth byte of process main word PMW 0 contains the decor extension number and is utilized when the system is in emulation mode.

Process main word PMW 1 is stored in bytes 4-7 of the process control block PCB. Details of PMW 1 is shown on Figure 10e. The status byte 1016 is the first byte in PMW 1 and stores the status register contents. The multiprocessor byte MP 1018 is significant in a multiprocessor architecture otherwise this field is zero. The second and fourth bytes of process main word 1 are the MBZ fields 1017 and 1019 respectively which must be zero for normal operation.

Process main word PMW 2 occupies bytes 8 through 11 of the process control block and is shown in more detail on Figure 10f. Referring to Figure 10f the field from bit 4 through bit 31 contains the local name SEG, SRA 1021 of the semaphore to which the PCB is linked when the process is either in the waiting or suspended states. The exception class and type field 1023 contains the class and the type of the interrupt-like exception which cause the process to enter the suspended state after an exception. The field from bits 4 through 15 is meaningless 1022 when a process is in a different state than those mentioned above.

Process main word PMW 3 occupies bytes 12 through 15 in PCB 400 and points to a decor extension table. Referring to Figure 10g for details of PMW 3 the DETSZ field 1024 defines the number of entries in the table and if this field is zero no decor extension is allowed to the process. The DETA field 1025 is the absolute address of the decor extension table in units of 16 bytes and is significant only if DETSZ is not 0. The decor extension table is made up of DETSZ entries. Each entry is one byte size. The DEXTth entry of the table defines the capability of the process to operate in the decor extension mode DEXT. When the DEXTth byte is 0 the decor extension number DEXT is not allowed, whereas if the DEXTth byte is 1 the decor extension number DEXT is allowed. Values of DEXT other than 0 and 1 are illegal. (See Figures 10a DEXT number 1004).

Bytes 16 through 23 of PCB 400 contains 2 address space words ASW 0 and ASW 1 respectively and each ASW contains a pointer to an array of segment table words. Both ASW 0 and ASW 1 respectively have the same format shown on Figure 10h. The size of the array of the segment table words is defined by the number of segment table words in an array and typically comprises six for ASW 0 and eight

for ASW 1. The STWSZ field 1026 indicates the size of the array of the segment table words. The segment table word array field STWA 1027 contains the absolute address STWA of the array in units of 16 bytes—i.e. the absolute address of the array is 16 times STWA in bytes.

Bytes 24 through 27 in the PCB contain an exception word EXW shown in greater detail on Figure 10i. The exception word contains a pointer (SEG, SRA) 1029 to an exception class table which defines the action to be taken following a process exception according to its class as stored in process main word PMW 2. (See Figure 10f). The MBZ field 1028 of exception word EXW must be 0.

The stack word SKW located in bytes 28 through 31 of the PCB contains the value of the top of the T register of the stack of the process when the process is not running and is shown in greater detail in Figure 10j. Referring to Figure 10j, bits 0 and 1 define the TAG field 1030. The TAG indicates the type of descriptor by its contents and must be zero for SKW. Bits 2 and 3 of the SKW word contain the RING field 1031 which contains the ring number associated with the segmented address of the stack for protection purposes and in this case must be zero. Bits 4 through 31 contain the segment number SEG, and the segment relative address SRA 1032 and is a field which identifies the segment described in a segment table and the segment relative address within the segment. The stack word SKW is updated every time the process leaves the running state. It is used to restore the T register contents every time the process becomes running. In this last case the TAG 1030 and RING 1031 are tested to be zero, otherwise an illegal PCB exception occurs.

Bytes 32 through 35 of the PCB 400 contain the instruction counter content word ICW sometimes also referred to as ICC. Referring to Figure 10k there are shown details of the instruction counter word ICW wherein the TAG field

1033 must contain binary 00 (i.e. values other than zero are illegal in the instruction counter). The current RING field 1034 occupying bits 2 and 3 defines the current ring number of the process to be used in determination of access rights to main storage. Bits 4 through 31 define the segment number and the segment relative address (SEG, SRA) 1035 which define the address of the next instruction to be executed.

The MBZ field in bytes 36 through 39 must be zero. (Note the MBZ field always indicates a field which must be zero). The MBZ word is tested every time the PCB is accessed from the name J, P. If it is not zero an illegal PCB exception occurs.

Stack base words SBW 0-2 occupy bytes 40-51 in the process control block 400. These words have the same format which is shown in greater detail on Figure 10 l. They are utilized during stack operations and whenever used their TAG field 1036 and RING field 1037 must be zero otherwise an illegal PCB exception occurs. Bits 4 through 31 contain the segmented address (SEG, SRA) 1038 of the first bytes of the stack segments for ring zero, 1 and 2 respectively.

Bytes 52 through 83 of the process control block 400 is a space reserved for the base registers saving area (8 words). Bytes 84 through 147 is a saving area which is utilized to save the values of all general registers (16 words). Bytes 148 through 179 is a saving area which is utilized to save the scientific registers (8 words).

Five double words are provided in the PCB 400 above the PCB zero address, for time accounting purposes when the accounting mode bit in the PMW 0 word is set. These words are located from PCB address minus 8 to PCB address minus 40. Each word contains a time or a time interval expressed in microsecond units in its first 52 bits with bits 52-63 filled with zeroes. The residual time out double word RTO (first 8

bytes above 0 in the PCB) contains the quantum of time which is actually spent by the processor on behalf of the process before a time out exception occurs. The RTO word is updated in the following way: each time the process exits the running state the process timer value is stored in the RTO word. Each time the process enters the running state, the process timer value is loaded from the RTO.

The running time accounting RUA double word at bytes 7 through 15 is a time counter which specifies the total amount of processor time a process was in the running state. The time accounted for is the time actually spent by the processor on behalf of the process exclusively. The RUA word is updated in the following way: each time the process exits the running state, the value of the process timer PT is read. The difference of the contents of RTO and PT is added to RUA. (Consecutively, the PT value is stored in RTO). Note that the time during which the process is suspended is not computed. The RTO and RUA words are updated even if the accounting mode bit is set to 0. However the CET, RTA, and WTA words (to be later described) are provided in the process control block only if the accounting mode bit in the process main word PMW 0 is set to 1. They are updated only in this case.

The waiting time accounting WTA word at bytes 17 through 23 is a real time counter which specifies the total amount of real time the process was in the waiting state. The WTA word is updated in the following way: each time the process exists the waiting state the time of day clock (not shown) value TOD is read and the Value of TOD minus the value of CET word is added to the WTA word.

The ready time accounting RTA word located at bytes 24 through 31 is a double word which is a real time counter which specifies the total amount of real time the process was in the ready state. The RTA is updated in the following way:

each time the process exits the ready state, the time of day clock value TOD is read, and the contents of TOD minus the contents of CET is added to RTA.

The current entry time CET double word at bytes 32 through 39 contains the time of day at which the process entered one of the following states: ready, waiting, running, and suspended.

SYSTEM BASE STRUCTURES:

Referring to Figure 6 the format of the system base 600 is shown. The system base resides in absolute main memory and is developed by firmware and is accessible via the boundary address register (BAR) which can be read but not written. The boundary address register BAR is below an area in main memory reserved for hardware and separates this area in memory reserved for hardware and the system base 600. Referring now to Figure 6 the system base 600 contains a number of system attributes which includes a job step number and a process group number (J, P) for the currently running process. From the logical name of the process J, P, the absolute address of the corresponding process control block PCB is obtained. The size and address of the J table are defined by the contents of the J table word (JTW). This word is located at the address defined by the BAR register. The format of the JTW is shown on Figure 11a. The size (JTSZ) 1101 or the J table 1204 on Figure 12 defines the number of entries in the J table 1204 which may be up to 255 entries. The JTSZ 1101 is an 8 bit positive integer; an out of J table exception occurs if J is greater than JTSZ. The absolute address of the J table 1204 is obtained by multiplying the J table pointer 1102 by 16. The J table 1204 contains J table entries whose format is shown in greater detail on Figure 11b. Each J table entry defines the absolute address of a P

table 1205 which is obtained by multiplying the P table pointer 1104 by 16. The size (PTSZ) 1103 of a P table defines the number of entries in the P table. The PTSZ is an 8 bit positive integer which may typically vary from 0 to 255 to indicate the number of entries in the P table. An out of P table exception occurs if P is greater than PTSZ. Each entry of the P table 1205 defines the absolute address of a process control block (PCB) 1206 by multiplying the process control block pointer 1107 by 16. A presence indicator P 1105 indicates the absence of a PCB 1206 when set to binary 0 and indicates the presence of a PCB when set to binary 1. (When the presence indicator P 1105 is found to be 0 a vacant P table entry exception occurs). Bits 1 through 7 of the P table indicator (Figure 11c) must be 0 (MBZ) 1106, otherwise an illegal P table entry exception occurs.

At address BAR plus 4 of the system base 600 there is the format byte of a G table word (GTW) shown in greater detail on Figure 11d. The size and the address of a G segment-table 1212 on Figure 1200 are defined by the contents of the G table word (GTW). The size (GTSZ) 1108 of the G table 1212 defines the number of entries in the G table which may typically be up to 255 entries. GTSZ is an 8 bit positive integer; an out of G table exception occurs if the G number is greater than the GTSZ. The absolute address of the G table 1212 is obtained by multiplying the G table pointer 1109 by 16. The format of the G segment table entry has a two word size (8 bytes) and is called a G segment descriptor. The format of the G segment descriptor is shown in detail on Figures 11e and 11f. All G segment descriptors are direct and therefore the indirect bit I, 1111 must be 0 otherwise an illegal segment descriptor exception occurs. The presence indicator P 1110 is a one bit field which when set to binary 1 indicates that a segment is defined in main storage for the segment number to which that descriptor corresponds;

whereas if it cleared to 0 no segment is defined and a reference to the segment descriptor causes a missing segment exception. The available bit A 1112 is a one bit field which indicates whether or not the segment is available; it is only checked if this segment is defined (i.e. P equals binary 1), otherwise it is ignored. The used flag field U 1113 indicates whether or not the segment has been accessed. If the U bit is set to binary 0 the segment has not been accessed; whereas if the U field is set to binary 1 the segment has been accessed. The written flag field W 1114 indicates whether the segment has been written. If W is set to binary 0 the segment has not been written; whereas if W is set to binary 1 the segment has been written. The gating indicator GS 1115 of a G segment descriptor must be set to binary 01, otherwise an illegal segment descriptor exception occurs. The reason for this is that a G segment always contains semaphores (although the reverse is not true i.e. all semaphores are not required to be in a G segment) and instructions on semaphores require the GS code to be binary 01. The absolute address of the base of a segment 1214 is defined in the G segment descriptor of Figure 11e by the 24 bit base field 1116; the content of this field is multiplied by 16 to obtain the absolute address. The second word of the G segment descriptor of Figure 11f occupies bit position 32 through 63 in the G table 1212. The RSU field 1117, bits 32 through 39 is reserved for software use and is generally ignored when used as a G segment descriptor as it is in this case. The MBZ field 1118 must be 0 otherwise an illegal segment exception occurs. Since the MBZ field 1118 occupies bits 40 through 51 it sets the SIZE field 1119 which is the field for a small segment SIZN; hence all G segments must be of the small segment type. The segment SIZN 1119 is a 12 bit positive integer defining the number of bytes in the segment and the segment size is interpreted as a multiple of 16. Therefore the segment size for

a G segment 1214 cannot exceed 2^{16} bytes (small segments).

Referring once again to the system base 600 of Figure 6 there are 9 system exception cell words located between BAR plus 8 and BAR plus 44. The format of the system exception cell words EXC is shown on Figure 11g. Since semaphores are utilized for transmitting messages to dedicated processes when a system exception occurs the pointers to these semaphores are found in 9 locations of memory each location called a system exception cell—one per class of system exception. The MBZ field 1120 must be set to binary 0 otherwise a system check occurs. Each exception cell (EXC) contains the system name G, D 1121 and 1122 respectively.

The channel exception cell located in BAR plus 44 of the system base 600 has a format which is similar to the system exception cell previously discussed and contains the system name GD of a semaphore which is used for transmitting messages to dedicated processes when a channel exception occurs.

An internal processor queue word IPQW is located beginning at BAR plus 48 and details of its format are shown on Figure 11h. The IPQW word points to the head of a queue of processes ready (Q/PR/RDY) as shown on Figure 9 by reference numerals 905 and 805. The queue of processes ready (Q/PR/RDY) links all processes which are in the ready state. It is referenced by the HEAD OF Q/PR/RDY-field 1124 (Figure 11h) of the IPQW word by pointing to the top of the ready process queue. The HEAD of Q/PR/RDY-field 1124 contains a 16 bit positive integer which is the displacement from the base of the G segment number 0, referred to as the GO segment, to the first byte of Q/PR/RDY. If this Q/PR/RDY bit field is 0, the ready queue is considered to be empty. The MBZ field 1123 must be 0 otherwise a system check occurs.

At BAR plus 52 of the system base 600 there is shown the storage for the initial and current retry counts whose format is shown in detail on Figure 11i. The NFS field 1125 is a nonfunctional storage field and is not utilized by the system base. The initial retry count field 1126 and the current retry count field 1127 are used to control the number of times automatic instruction retry is executed before a machine error is made to produce a machine failure exception condition. They are loaded with the same number by a Reset Retry Count (not shown herein).

The running process word (RPW), shown in Figure 11j, is stored in BAR plus 56 of the system base 600 and is used to store the name of the running process with its priority in case of a monoprocessor architecture. The NFS fields 1128 and 1131 respectively are nonfunctional storage fields and may be utilized for any purpose by any facility but is generally not utilized by the system base. The priority level of a running process is stored in the PRI field 1129. An asynchronous trap bit is stored in AB field 1130; whereas an asynchronous trap ring is stored in ARN field 1132. The logical name J, P of the running process in case of a monoprocessor architecture is stored in the J, P field 1133.

An Absolutization Table Pointer word shown on Figure 11k is located at BAR plus 60 in the system base 600 and is utilized in initial system load to initialize the absolute addresses in the initial system load (ISL) program by adding the contents of BAR to all absolute addresses in the ISL program. The Absolutization Table Pointer 1135 defines the location of an Absolutization Table (not shown). The Absolutization Table Size is shown by the ATSZ field 1134.

The CPU serial number word shown on Figure 11l is a 4 byte word located at BAR plus 64 and contains the serial number of the CPU in the CPU serial number field 1136.

A main storage upper limit word shown on Figure 11m is located at BAR plus 68 and indicates the main storage upper limit 1139 by providing the absolute address of the last available word in main storage.

At BAR plus 72 there is located a word shown on Figure 11n for providing the initial system load ISL device channel number (CN) 1140 and the hardware device channel number (CN) 1141.

The type and subtype of a device used in the computer system is shown by a hardware device type word (Figure 11o) in fields 1143 and 1144 respectively; where the RSU field 1142 is reserved for software. This word is found in the system base at BAR plus 76. A similar word having a similar type format shown on Figure 11p contains the type and subtype of the device used in the initial system load. This word is located at BAR plus 80.

When the restart button of a computer is pressed, a simulated V-operation is performed on a semaphore and the Ready state is entered. A pointer to this semaphore is found at BAR plus 84 of the system base 600 and is called a restart cell word, and has a format shown on Figure 11q. The format is similar to the system exception cell described supra and contains the system name G, D of a semaphore in the G field 1149 and D field 1150 respectively. The MBZ field 1148 must be 0.

When there is more than one processor to the computer system, a word is provided in the system base 600 at BAR plus 88 for multiprocess extension. Details of this word are shown on Figure 11r.

Examples of System Base and Process Control Block Use

Referring to Figure 12 there is shown one example, how the system base may be utilized in combination with the process control block in order to address and access a user

segment, a system segment, or a queue of processes read (Q/PR/RDY) segment. Main memory 1200 has a portion 1203 reserved for hardware use. A boundary address register BAR 1202 separates the system base 1215 from the portion of memory 1203 reserved for hardware. The boundary address register BAR 1202 is utilized to address items in the system base 1215 by adding the contents of the boundary address register to the displacement in 4 byte units of the item desired in the system base. This address then points to the first byte of the item in the system base desired. In Figure 12 the BAR 1202 is pointing at the J table word (JTW). The J table word, as previously discussed, has a pointer which points to a J table 1204. By indexing to the J number shown on Figure 5, a J table entry 1216 is obtained. At the J table entry there is a P table pointer which points to the absolute address of the P table 1205. By indexing to the P number (see Figure 5) within P table 1205 the absolute address of the process control block 1206 is obtained. As previously shown in process control block PCB 1206 there are two address space words ASW 0 and ASW 1. The high order bits of the segment table number field STN in the base register 1201 is used to access one of these two address space words, in this instance ASW 1 which has a segment table word array STWA pointer that points to segment table word array STWA 1208. Together with the segment table number STN of the base register 1201 one of 8 segment table words is accessed in STWA 1208, which points to one of 8 segment tables 1210. The segment table entry STE from base register 1201 is then utilized to make one of 256 entries in segment table 1210 where a segment descriptor is located. The segment descriptor is then utilized to access a user segment 1211. (For greater detail see Application No. _____ filed on _____ (entitled Segmented Address Development and assigned to the same assignee as the instant invention and is hereby incorporated by reference herein).

In order to access a system segment 1214 which is utilized to store semaphores a G table word GTW is utilized in the system base 1215. The address of the G table word is obtained by adding the displacement of the G table word in the system base to the boundary address register BAR 1202. (See Figure 6). The G table word GTW includes a G table pointer which points to a G table 1212. By utilizing a G number available to the system and indexing in the G table a G segment descriptor is accessed which is utilized to address a system segment 1214.

Similarly the system base 1215 is utilized to access the queue of process ready (Q/PR/RDY) 1213 by locating an internal processor queue word IPQW which points to the Q/PR/RDY segment 1213.

Control Unit

Referring to Figures 13a-13c details of the control unit are shown. The control unit, although shown separate from the central processing unit (CPU), is in actuality a part of the CPU and is comprised of a control store unit CSU 1301, a control store interface adapter CIA 1302 and appurtenant subunits, control store loader CSL 1303 and control and load unit CLU 1304.

The control store unit CSU 1301 receives micro-instructions from the control store loader CSL 1303 via the control and load unit CLU 1304 and the control store interface adapter CIA 1302. Under normal operating conditions, microprograms are loaded from an external source during system initialization and become a permanent control function of the machine. However the control store unit CSU 1301 has the ability to be reloaded and initialized in a manner that provides for a variety of central processing unit CPU 1306 operational modes. The

following modes of operation of the CPU are available under control of the CSU 1301; (a) native mode; (b) emulation mode; (c) concurrent native and emulation modes; (d) diagnostic mode. This capability is possible because the micro-instructions resident in the CSU are the source of micro-operations used to control the operation of all other CPU functional units such as the emulation unit 1316, the arithmetic logic unit ALU 1317, the instruction fetch unit IFU 1318, the address control unit ACU 1319 and the data management unit DMU 1321. Also shown within the central processing unit CPU 1306 are previously described general registers 1307, base registers 1308, scientific registers 1309, T-registers 1310, status registers 1311, instruction counter IC 1312, and hardware control mask register 1313.

Typically the control store unit CSU 1301 is a 9K bipolar integrated circuit programmable read-only memory (PROM) mixed with read/write random access store (RAM). It has a typical 150 nanosecond read cycle and a 450 nanosecond write cycle. Each location of control store stores one 84-bit micro-instruction word (to be later more fully described), and each micro-instruction word controls one CPU cycle. As each location of the control store of the control store unit CSU 1301 is read, its contents are decoded by micro-operation decoders which provide micro-operation control signals each of which causes a specific operation within the CPU to take place (to be later described in detail).

By grouping locations within each micro-instruction word (to be later described in detail) control store sequences are obtained that can perform a specific CPU operation or instruction. As each instruction is initiated by the CPU, certain bits within the op-code are used to determine the control store starting sequence. Testing of certain flops (not

shown) which are set or reset by instruction decode functions allows the control store memory to branch to a more specific sequence when necessary.

The control store interface adapter CIA 1302 communicates with the control store unit 1301, the data management unit DMU 1321, the address control unit ACU 1319, and the arithmetic logic unit ALU 1317 for directing the operation of the control store memory 1333 of Figure 13b. The CIA 1302 includes logic for control store address modification, testing, error checking, and hardware address generation. Hardware address generation is utilized generally for developing the starting address of error sequences or for the initialization sequence.

The data management unit DMU 1321 provides the interface between the CPU 1306 and the main memory and/or buffer store memory shown on Figure 1. It is the responsibility of the data management unit to recognize which unit contains the information required by other units and strobe the information into the CPU registers at the proper time. The data management unit DMU also performs the masking during partial write operations.

The instruction fetch unit IFU 1318 interfaces with the DMU 1321, the ACU 1319, the ALU 1317, and the CSU 1301, and is responsible for keeping the CPU supplied with instructions. The instruction fetch unit has the next instruction available in its registers before the completion of the present instruction. To provide this capability, the instruction fetch unit IFU 1318 contains a 12-byte instruction register (not shown) that normally contains more than one instruction. In addition, the IFU, under control of the CSU, requests information (instructions) from main memory before the instruction is actually needed, thus keeping its 12-byte instruction register constantly updated. Instructions are thus prefetched by

means of normally unused memory cycles. The instruction fetch unit also decodes each instruction and informs the other units of the instruction's length and format.

The address control unit ACU 1319 communicates with the IFU, ALU, DMU, and the CSU via the CIA. The ACU 1319 is responsible for all address development in the CPU. All operations of the ACU, including transfers to, from, and within the unit, are directed by CSU micro-operation and logic in the unit. The normal cycling of the ACU depends on the types of addresses in the instruction rather than on the type of the instruction. Depending on the address types the ACU may perform different operations for each address in an instruction. The ACU also contains an associative memory 1319a that typically stores the base address of the 8 most recently used memory segments, along with their segment numbers. Each time a memory request is made, the segment number is checked against the associative memory contents to determine if the base address of the segment has already been developed and stored. If the base address is contained in the associative memory 1319a, this address is used in the absolute address development, and a considerable amount of time is saved. If the base address is not contained in the associative memory 1319a it is developed by accessing the main memory tables. However, after the base address of the segment is developed, it is stored in the associative memory, along with the segment number, for future reference.

Interfacing with the ACU, IFU, DMU, and the CSU is the arithmetic and logic unit ALU 1317. Its primary function is to perform the arithmetic operations and data manipulations required of the CPU. The operations of the arithmetic logic unit are completely dependent on micro-operation control signals from the control store unit CSU 1301.

Associated with the ALU 1317 and the CSU 1301 is the scratch pad memory unit LSU 1315, (sometimes referred to also as the local store unit). It is typically comprised of 256-location (32 bits per location) solid state memory and selection and read/write logic for that memory. The scratch pad memory 1315 is used to store CPU control information and maintainability information. In addition, the scratch pad memory 1315 contains working locations which are primarily used for temporary storage of operands and partial results during data manipulation. Also associated with the ALU 1317 is an auxiliary memory 1317a comprised typically of 64 flip-flops for storing miscellaneous states of the computer system.

The CPU also has a clocking unit 1320 and is essentially 2 clocking systems in 1: the first clocking system generates the timing for the control interface adapter CIA 1302 and the second clocking system generates the timing pulses for the operations of the functional unit within the central processing unit.

Referring now to Figure 13c there is shown the format of the control store word 1325. The control store word is typically 84 bits wide and is divided into 6 main fields:

a. sequence type field 1326 (3 bits);
b. branching and/or micro-operations 1327 (23 bits);
c. constant generation and designation 1328 (14 bits);
d. data to bus 1329 (8 bits);
e micro-operations 1330 (32 bits); and
f. checking 1331 (4 bits).

The 3-bit F field of the control store word 1325 is used as a sequence control field. There are typically 7 different sequence types and 1 reserved type for the instant computer system. Referring to block 1335 of Figure 13b, when E field equals binary 0, 1, or 2, the branching field A, B, C, D, and I, of micro-instruction 1325 is utilized to generate the

next address. The first 6 bits of KS register 1337 is utilized together with the B field, a C test results, the D test results and the L field to provide the next address of the next micro-instruction which is then placed in address register KS 1337. When the E field is set to binary 4 (see block 1335) the next address selected is taken from interrupt return register KA 1339. The address stored in the KA register is the one generated by the next address generation logic when the hardware interrupt occurs. When the E field is set to binary 5 a branch is used to initiate a subreturn from a micro-program subroutine. When used, the contents of the return register KR 1346 are used as the next control store address. The return register 1346 is loaded by issuing a control store command which will load present control store address in KS register 1337 plus 1, from incrementor 1338, into the KR register 1346. A one-level-nesting subroutine ability is provided via the KT return branch register 1347. Every time the KR register 1346 is loaded the old contents of the KR register is transferred to the KT register 1347 every time the micro-program return is called; the contents of the KT register will transfer to the KR register. Third level nesting subroutine ability is provided by the KU register 1340; and fourth level nesting subroutine ability is provided by the KV return branch register 1349. When the E field of the control store word is set to binary 6 the next control store word addressed is equal to the present address in KS register 1337 plus 1 in incrementor 1338. When the E field is set to binary 7 the CSU 1301 enters the diagnostic mode and the next address will be the present address plus 1.

In addition to the sequencing control of branching to the next control store address described above and shown in block 1335, there is hardware generated sequence control shown in block 1336 of Figure 13b. (Note: Blocks 1335

and 1336 are in actuality hardware registers drawn so as to depict the different forms that the microinstruction words may take.) The hardware generated branches are overriding conditions (such as errors, initialize, control store scan, etc.) which surpass the E field and force a fixed address into the control store address register KS 1337. The branch is made by forcing an interrupt line high (not shown) for one clock period and storing the address which would have been generated under the control of the E field into the KA interrupt return register 1339. A hardware generated address will be placed into the control store address register. Certain hardware/firmware generated interrupts take priority when the interrupt-block flip-flop (not shown) which prevents additional interrupts in their class from being executed until the interrupting condition has been satisfied. A firmware micro-operation exists for controlling the resetting of the interrupt-block flip-flop for those sequences which are under firmware control. Those sequences under hardware control automatically generate a reset of the block-flop at the end of the sequence. The following conditions, listed by priority, exists in this category; (a) control store load; (b) control store scan; (c) hardware error; (d) software error. The remaining hardware conditions do not set the interrupt block-flop but do cause an immediate action to occur when generated. The following conditions listed by priority, exist in this category;

(a) initialize;
(b) soft-clear;
(c) enter maintenance panel;
(d) enter maintenance panel;
(e) hardware exit.

An initialize signal causes the CSU 1301 to branch to address binary 0, clear hardware resettable errors and execute a control store load operation followed by a control

store scan sequence under hardware control. It will also perform system initialize. A soft-clear signal causes the CSU 1301 to branch to address binary 0, clear hardware resettable errors and reset the interrupt block-flop. An enter maintenance panel signal causes the CSU to branch to the address preset in the CSU address switches on the maintenance panel (not shown).

An enter maintenance channel signal causes the CSU to branch to an address generated via the maintenance channel (not shown). The address loaded is from maintenance bus QMB 1344, which is part of the maintenance channel, and is right-justified. A hardware exit signal causes the CSU to branch to binary address 2. This sequence is used as a maintenance facility. At the end of the sequence a return is intiated by issuing an E field branch with the E field set to binary 4.

A control store load signal causes the CSU to branch to address binary 0. It also turns off the CSU read-cycle flop (not shown), the system clock 1320, and places the CSU in the load state. In the load state the CSU can be loaded from the control store loader CSL 1303, the IOC 1305, the main memory 102, or the maintenance panel 1355. When loaded from the CSL an automatic scan is generated at the end of the load. When loaded from any other media a scan may be issued by either generating a micro-operation signal or setting the scan switch on the maintenance panel. A control store scan signal causes the CSU to branch to an address binary 0. A control store scan is under hardware control for the duration of the sequence. During the scan the system clock 1320 is off and therefore no commands or tests are executed. At the end of the scan sequence the hardware transfers the contents of the interrupt return register KA to the address register KS, the system clock is turned on and control is returned to the firmware.

A hardware error signal causes the CSU to branch to address binary 4. In the normal processing mode a hardware error detected in any CPU functional unit will activate a hardware error line (not shown). The control store sequence generated will test the system conditions to determine the action to be taken. In the diagnostic mode, error conditions which are hardware detectable are visible to microdiagnostics. The microdiagnostics control the action to be taken. A software error signal on the other hand causes the control store to branch to address binnary 1. This address is the start of the software error reporting sequence which is under micro-program control.

Referring once again to Figure 13c the E field 1326 is a 3 bit field for the branch code as previously described. The branching and/or micro-operation field 1327 is comprised of the A, B, C, D, and L fields (also shown on block 1335 of Figure 13b) wherein the A field is the upper 6 bits of the next address, the B field is the middle 4 bits of next address of the mask field 64 tests, the D field is another 6 bit test field for 1 of 64 tests, and the L field is the least significant bit. The K field 1328 is a 14 bit field of which 6 bits are for the constant field, 4 bits are for a constant or steering field, and 4 bits are a steering field for a constant. The data to bus field 1329 is comprised of the QA field having 4 bits for controlling information to the QA portion of the QMB bus 1344 and the QB field has 4 bits for controlling information to the QB portion of the QMB bus 1344. The F field 1330 is a 32 bit field which is coded to generate micro-operation subcommands. The P field 1331 is comprised of 4 bits reserved for checking.

In operation the micro-instruction words are stored in the control store array 1333. During a cycle of operation, the control store array is addressed by the contents of the KS address register 1337. This causes the contents of the

location specified by the address to be read into the group of read latches 1357. Portions of the word contents of the read latches are distributed or transferred to storage registers within each of the functional units in the CPU. Each functional unit includes decoding logic circuits for generating the requisite subcommands specified by the control store word under control of the system clock source. In general decoding is performed within each functional unit in the CPU rather than being performed centrally in order to minimize the decoding time and to reduce the number of cables which would be normally required for transmitting command signals if decoding were performed centrally. Additionally, the decoding is done within each unit to avoid timing problems arising from differences in cable delays. Further, by decoding subcommands with each unit, those signals which are representative of certain conditions existing within the functional unit are required for the generation of certain subcommand signals do not have to be returned to the CIA unit 1302. A typical decoder unit 1359 is shown in Figure 13b as receiving various fields from micro-instruction words and generating micro-operation signals a, b, c, d, . . . q, r. A typical micro-operation decoder 1359 receives commands from a micro-instruction word. The field from the micro-instruction word is decoded and sets one of a plurality of lines s, t, u. . . . y, z high. A matrix is formed by having predetermined control line impedance coupled to the s-z lines at points α, β, $\gamma \ldots \psi$, ω. Typically then when the field from a micro-instruction is decoded one of the lines s-z goes high. Since the black dots shown in the matrix by Greek letters α through ω represent impedance coupling between the two sets of lines, any electrical signal propagating along any horizontal wire will be coupled through to propagate along the vertical wire a-r where an impedance coupling (black dot) is indicated.

Each vertical line a-r then may be coupled as one input to one each of AND gates 1360-1365. Other input signals may also be coupled to AND gates 1360-1365. Other input signals may also be coupled to AND gates 1360-1365 including a timing signal t_s from the central timing unit. Accordingly as each timing signal t_s goes high those gates which have all the other input signals high will be enabled and will provide micro-instruction signals to predetermined functional units in the CPU. For example, if a command 1341 from read latch 1357 is decoded and a horizontal line is high the a, b, c, and q vertical control lines will be high and AND gates 1360, 1361, 1362, and 1364, will be enabled as the t_s timing signal is applied to these gates sequentially. Hence the combination in which the vertical control lines are coupled to the horizontal control line at different points represented by the Greek letters α through ω represent a permanent switching matrix for supplying micro-operation signals to the central processing unit CPU for controlling the functional units within the central processing unit by micro-instructions furnished from the control store array 1333. Thus permanent firmware having an alterability feature can be built into the machine of this invention by merely specifying the sequence of micro-operations that is required as a capability of the computer system.

Under normal conditions data is written into the control store array 1333 via the CPU write-data register also known as the local register YO 1343. A control flop (not shown) defines whether the upper half or lower half of the storage array is to be written. Data from the control and load unit CLU 1304 arrives at the CIA/CSU via the maintenance bus QMB 1344 and is buffered by the storage local register YO 1343 before being written into the control store array 1333. The storage local register 1343 is time

shared as both a read and a write local register. The multiplexor KQM 1345 may be controlled by either the maintenance panel 1355 or by micro-diagnostics and provides a read out path from the registers connected to it. A compare register KP 1350 is provided for non-functional use and is used mainly for maintenance purposes and is utilized along with compare logic 1352 and decode logic 1351.

DISPATCHER FIRMWARE FOR CONTROL OF PROCESSES

The dispatcher is a firmware/hardware unit whose main purpose is to manage the various queues of processes, and to switch between processes and includes the updating of the queues of processes, the process control blocks PCB's, the running process word in the system base and the register of a new process. It also delivers messages to a process which is actually waiting for it on a semaphore (after a V-operation, a simulated V-operation for the IOC or for the exception handler). It also enqueues a message on a semaphore after a P-operation which freed a message link when a process is waiting on the free-link semaphore to deliver its message.

The dispatcher unit further calls on the native mode instructions firmware after the "rolling-in" of a process executing in native mode, or after a "contest" if the current process remains running and is executing in native mode. It also calls on decor extension firmware for:

(a) a temporary call during the rolling-out of a process executing in that decor extension;

(b) a temporary call during the rolling-in of a process executing in that decor extension;

(c) a definitive call at the end of the rolling-in of a process executing in that decor extension; and,

(d) a definitive call after a contest if the current process remains running and is executing in that decor extension.

Moreover the dispatcher places the system in the idle loop when there is no running process.

There are several ways of entering or leaving the dispatcher as follows:

(a) The initialization procedure (SIP) provides entry as the last step, (See Patent Application No. _____ entitled System Initialization Procedure, assigned to the same assignee named herein.)

(2) The Start and Suspend Instruction provides for the entry into the dispatcher. The Start instruction starts a process and the Suspend instruction terminates a process. (See Patent Application No. _____ entitled Start and Suspend Instruction.)

(3) The P and V operations provide for entry into the dispatcher. The P op picks up a message from a semaphore and if no message then the process goes into wait state. (See Patent Application No. _____ entitled "P and V Operations of Semaphores".)

Briefly then, the dispatcher is the main mechanism which manages the processes and accordingly the process control blocks PCB's by deciding which process is to be run and then taking the appropriate action, such as rolling-out the currently running process (i.e. write into the PCB all information relative to the currently running process contained in hardware registers, scratch pad memory, etc.), and rolling-in the new process (i.e. write from the PCB to various hardware registers, scratch pad memory, etc., all information necessary to run the new process.)

The representation of the functions performed by the dispatcher are shown in flow-charts Figs. 14a-14i. For

example, block 1402 of Fig. 14a is a representation of the function performed by the dispatcher wherein a micro-program word is delivered by the control store unit, and upon decoding by decoder 1359 controls the applicable portions of the CPU via an appropriate series of micro-operation signals 1360, 1361, etc. to retrieve the IPQW from the system base in storage subsystem 102, and transfer it to scratch pad memory 1315. Simultaneously the dispatcher fetches 1404 the GO segment descriptor (see Figure 12) from the G table of segment descriptors pointed to by the G table word GTW in the system base. Bits 16-31 of the IPQW word contain a 16-bit positive integer which is the displacement from the base of the G segment number 0, referred to as the GO segment, to the head (first byte) of the Q/PR/RDY, queue of processes ready. If bits 16-31 of the IPQW word is 0, 1403 the ready queue is considered to be empty. If the ready queue is empty it indicates that there is no process currently waiting in the Q/PR/RDY and the ready-queue is empty. The next question to be decided in decision block 1405 is whether or not a process is currently running in the machine by determining whether or not the Vacant Indicator is set. (The Vacant Indicator is a flip-flop located in auxiliary memory 1317a; it is set when there is no current process CJP running in the processor). If the Vacant Indicator is set (i.e. there is no process currently running) and since it was previously determined that there is no process in the ready-queue waiting to use the processor then the machine goes in the idle state 1406. However if there is a process currently running in the machine, but there is no one waiting to use the machine, the current process accesses its next instruction 1407.

Returning now to decision block 1403, of the flow chart of Fig. 14a if there is a positive integer in the pointer area of the IPQW (ie. bits 16-31) the head of the ready queue

pointed to by the IPQW word in the GO segment is fetched to the scratch pad memory. (Note: To avoid repetition and in the interest of clarity the intermediate functions of the dispatcher in conjunction with the control unit and CPU will be omitted; however it is to be understood that such intermediate functions as previously described by example are present typically.) Up to this point it has been determined that there is some process waiting in the ready-queue. Before further action can be taken it is necessary to determine if there is a process currently running in the central processor. This is determined at decision block 1410 of the flow chart representation and if there is no process currently running in the central processor (i.e. no CJP) the head of the ready queue is run 1412. However if there is a process running in the central processor the dispatcher must determine who has priority — the currently running process or the head of the ready queue? Accordingly therefore the priority byte of the current process (CJP) which is located in the running process word of the system base 600 or in process main word 0 PMW 0 of PCB 400, is fetched 1413. A decision is then made 1414 as to whether or not the current running process CJP is of lower priority than the new process NJP waiting at the head of the ready queue? (See decision block 1414.) If CJP is not of lower priority than NJP, CJP remains in control of the central processor and the contest indicator is reset 1415. (The contest indicator is always set to 0 except when one or more new processes have been placed on the ready-queue since the beginning of the last instruction executed on behalf of CJP thereby creating the possibility of a conflict; under these conditions the contest indicator is set to binary 1). Before the current process CJP is permitted to continue however, and execute further instructions a determination is made whether or not the CJP is running in decor extension

mode 1415. If CJP is running in decor extension mode then the next instruction is executed in emulation mode (i.e. decor extension) and if it is not running in decor extension mode then the next instruction is executed in native mode. Returning once gain to decision block 1414, if the NJP at the head of the ready queue has a higher priority than the CJP (i.e. its priority number is lower than CJP's priority number, the current running process CJP is "rolled-out" of the machine and the new process NJP is "rolled-in" to the machine. Accordingly a firmware priority subroutine PRIQ 1418 directs the enqueueing of the current process CJP into the ready-queue by LIFO priority and by priority number by first "rolling-out" the CJP under the direction of the firmware subroutine RLLO 1419. The RLLO subroutine directs the writing of the information of the CJP that is stored in the general registers, base registers, scientific registers, T-register, status registers, and instruction counter, back into the appropriate storage areas of the process control block PCB in main memory, and it directs the updating of the RUA. Moreover the DEXT number of process main word 0 (PMW0) in PCB 400 is updated 1420.

The new process NJP is now ready to be "rolled-in". The boundary address register BAR is fetched 1422 and the running process word RPW is fetched from address BAR plus 56 of the system base. See block 1423. The name of the new process NJP is next written into the running process word RPW and since the name of the new process NJP was written in the process link PL of the Q/PR/RDY, the name in the process link PL is therefore now placed in the RPW, block 1424. Therefore NJP from the ready-queue now becomes CJP and is entitled to control the central processor and accordingly is no longer waiting in the Q/PR/RDY and must be dequeued by taking its name out of the process link

PL of the Q/PR/RDY, block 1425. When this is done the queue of processes ready Q/PR/RDY is updated by the firmware subroutine UQLK, 1425a. Accordingly the JP number of the process which was just taken out of the machine is placed in a process link in the Q/PR/RDY, since it now no longer has control of the machine and must wait for it, 1426. At this point the transformation of giving this control of the central processor to the new process and placing the old process in a ready queue is accomplished and because there is a process (the new CJP) in control of the central processor, the Vacant Indicator is set to 0, 1427; if on the other hand there were no CJP in control of the central processor the Vacant Indicator would be set to 1. At this point the assignment of the processor is complete and a new process has acquired the central processor whereas the old process was placed in a ready queue; however, the new process is not ready to run yet for the hardware of the central processing unit, 1306 of Figure 13a, such as the general registers 1307, the base registers 1308, scientific registers 1309, the T-register 1310, the status registers 1311, and the instruction counter 1312, must be supplied with control information from the process control block of the new process.

Accordingly the firmware subroutine 1430 first fetches PMW 3 from the PCB (Figure 4) to scratch pad memory 1315, and then fetches PMW 0. The MBZ field of PMW 0 is checked 1433 and if it is not binary 0 an illegal PCB exception results. However if the MBZ field of PMW 0 is zero then PMW 1 is fetched 1434. Again the MBZ field of PMW 1 is tested to determine whether or not it is a binary 0. If it is not binary 0 there is an illegal PCB exception; whereas if it is equal to 0 the dispatcher proceeds to C.

Accordingly address space word 0 ASW 0 is fetched from the appropriate space in the PCB and the segment table

word size STWSZ is tested 1437 to determine whether or not it is less than 7. If it is greater than 7 an illegal PCB results if it is less than or equal to 7, then ASW 1 is fetched from the PCB, block 1438, and its STWSZ field is tested 1439 to determine whether or not it is less than or equal to 8. If its field is greater than 8 then an illegal PCB results; however, if its STWSZ field is equal or less than 8 the exception word EXW is fetched 1440, and its MBZ field is tested to determine whether or not it is equal to 0. If its MBZ field is not equal to 0 an illegal PCB results whereas if it is equal to 0 the stack word SKW is fetched 1442, and its MBZ field is tested 1443 to determine whether or not it is equal to 0. If the MBZ field is not equal to 0 then an illegal PCB results; whereas if it is equal to 0 the instruction counter word ICW is fetched from the PCB and placed in the instruction counter IC and its TAG field is tested to determine whether or not it is equal to 0, 1445. If the TAG field is not equal to 0 then an illegal PCB results. However if the TAG field is equal to 0 the MBZ word is fetched 1446 and its MBZ field (bits 0-31) is tested to determine whether or not it is equal to 0, 1447. If it is not equal to 0 then an illegal PCB results; whereas if it is equal to 0 stack base words 0, 1 and 2 SBW 0, 1 and 2 are fetched 1448. The contents of the 8 base registers in the base register saving area of the PCB is then fetched 1449 and stored in the machine's base registers 1308. Then the contents of the 16 general registers from the general register saving area of the PCB are fetched 1450 and stored in the machine's general registers 1307. Before fetching the contents of the scientific registers however a check is made of the capability byte of process main word 0 (PMW 0) to determine whether or not scientific mode is being utilized 1451. If scientific mode is being utilized then the contents of the scientific registers from the scientific register saving area of the PCB are fetched and stored 1452. The firmware then

proceeds to make a check of the capability byte of PMW 0 to determine whether or not the accounting mode is being utilized 1453. If the accounting mode is being utilized (i.e. accounting bit of capability byte set to binary 1) the accounting words exists in the PCB and the ready time account word RTA is updated. Then the firmware proceeds to determine whether or not the DEXT number is set to 0, 1454. If it is not set to 0 it indicates that the machine may be in emulation mode (i.e. the decor extension capability is being utilized) and accordingly the DEXT number of PMW 0 is checked 1455 to determine whether or not it is greater or less than the DETSZ field of process main word 3 and if it is greater than the DETSZ field an illegal PCB exception 1456 results since the DEXT number is less than the DETSZ field but not equal to zero the machine is performing in legal emulation mode and proceeds to F. Returning to decision block 1454 if the DEXT field is binary 0 then native mode is being performed and the machine fetches STW's, 1457. The residual time out word RTO word of the PCB is fetched 1458 and the process timer is loaded, with the time limit that the CJP may spend in the running state.

Up to this point either (a) a new process NJP has been "rolled-in" to take control of the CPU when there was an old process CJP in the machine and the new process NJP was at a higher priority than the old process CJP or (b) there was no CJP in control of the CPU and the head of the ready queue was rung. Briefly under condition (a) the CJP was taken out of the RPW and placed in a process link PL in the Q/PR/RDY, and the NJP in a process link PL in Q/PR/RDY was placed in the RPW thus effectively switching the positions of the 2 processes giving control to the NJP which now becomes the CJP and taking control away from the old CJP. Then the PCB of the NJP was accessed and the information required to run NJP (now the

CJP) was placed in scratch pad memory or the array of registers in the ACU.

If there was no CJP in control of the CPU (condition b), then the head of the ready queue was run—i.e. the NJP became the CJP because of the dispatcher taking the NJP from the process link PL from the head of the ready queue and placing it in the RPW word. By doing this a process link PL was left empty in the Q/PR/RDY and it is required to take it out.

Accordingly beginning now at decision block 1461 the firmware determines whether or not there was a CJP in control of the CPU and if there was a free process link (FPLS) it was accessed and enqueued and the CJP was written therein. However if there was no CJP in control of the CPU state byte of PMW 0 of the NJP is updated 1460 and again there is a determination whether or not there was a CJP in the machine 1463. If there was no CJP in control of the processor the process link of the NJP (which was in the Q/PR/RDY and is now in control of the machine) is taken out of the Q/PR/RDY 1466 (i.e. dequeued from the Q/PR/RDY) and becomes a free link semaphore FLSP and is now enqueued in the free process link queue (805 on Figure 9) and becomes a part of the free process link queue 1466a. The contents of the boundary address register BAR are fetched 1464 and the running process word RPW of the NJP (now the CJP) located at BAR plus 56 of the system base is updated by placing the NJP identification in the RPW, block 1465. The Vacant Indicator is set to 0 in the case where there had been no CJP. Next the Contest Indicator (i.e. a flip-flop in auxiliary memory 1317a used to indicate a possible conflict in priority between the CJP and a process placed in the ready queue) is set to 0, 1467 and the Segment Associator (AS 132 on Figure 1) which is a typical content addressable memory is cleared, 1471; then the

process mode is entered 1470. (A process mode indicates that exceptions are handled by the process which is operating in the processor rather than by the operating system). The firmware then continues to CAB 1480 and the asynchronous trap bit AB is checked to determine whether or not it is set to binary 1, 1481. If the AB bit is set to binary 1 then a check is made 1482 to determine whether or not the process ring number PRN is greater than or equal to the asynchronous trap bit ARN. (The AB and the ARN are located in the priority byte of the PCB of each process and is meaningful when the process is in the running state; the AB and ARN are obtained from the RPW located at BAR plus 56 of the system base). The AB and ARN in BAR plus 56 of the RPW are reset since the next step 1484 is to proceed to an asynchronous trap routine which would take care of the conditions that cause the asynchronous trap bit or the asynchronous ring number to be set in the first place, and if these were not reset then at the next pass by the firmware it would give an indication that something was wrong where actually there was nothing wrong, and hence would always proceed to the asynchronous trap routine 1484 and would never execute. Going back now to decision blocks 1481 and 1482 if the AB bit is not set or AB bit is set and PRN is not greater than ARN, then the firmware would proceed to determine in what mode the processor would run—in the normal mode or in the emulation mode. Accordingly the DEXT number is checked to determine whether or not it is set to 0 and if it is set to 0 the normal mode of the machine is run 1487. However if the DEXT number is not set to 0 the emulation mode is run 1486.

The Switch System Base Instruction

Referring now to Figure 15a there is shown the structure of the switch system base instruction. The operation code OP

1501 is similar to the operation code of any instruction and indicates the operation to be performed. The complementary code C 1502 is used in conjunction with the op-code 1501 when the op-code shares more than one instruction 1501 and the complementary code 1502 determines the precise instruction to be performed. For example, the P and V operations have 12 instructions and they will use one op-code. The complementary code together with the op-code determine which of the 12 P and V instructions is to be performed. The address syllable AS 1503 of the switch base instruction is the address pointing to a new copy of the system base which is to be loaded into main memory and also scratchpad memory. Accordingly, this address points to an entirely new system base. Typically a 60 byte zone, starting at the address specified by the address field of the switch system base instruction, is transferred into the system base area which starts at address BAR. This transfer is also made into the scratchpad memory. This zone, therefore, is the new system base which replaces the previous system base. The running process, therefore, disappears because its PCB is not updated and no link manipulation is performed. The processor is then placed in the VACANT state, and the dispatcher entered.

It is the system software responsibility to prepare the new information for the system base and the new GO segment before executing the SWSB instruction. (See applications included by reference herein numbers 12-16). For that reason, the SWSB instruction must be entered with the hardware gate HG locked, otherwise it returns a condition code CC without taking further action. Upon completion of SWSB instruction the HG will be left unlocked. Also upon completion of the SWSB instruction the vacant switch is set to 1. This indicates no process is running and as a result the system will go to the head of the ready queue for the internal

process queue word IPQW and pick out the pointer to the queue of processes ready. It will then proceed to the head of the ready queue and start up a new process. The process which executed the SWSB instruction will terminate because a vacant condition has been placed in the vacant indicator, which means that there is no running process.

Referring now to Figures 15b and 15c the SWSB instruction in firmware will be described in greater detail. The SWSB instruction starts 1504 and a determination is made 1505 whether or not the process ring number PRN is 0. This is necessary so that the SWSB instruction executes with the highest privilege. If the PRN is not 0 then a privileged instruction exception 1506 occurs. If on the other hand PRN is equal to zero the hardware gate located in auxiliary memory 1317a of the ALU 1317, is checked 1507 to determine whether or not it is locked. It is important that the HG is locked during the execution of the SWSB instruction. When the HG is locked no I/O interrupts or exceptions are permitted. If such exception or an error condition occurs while the HG is locked the system will crash. The system crash is a more desirable alternative while the system base is being changed, then permitting the interrupt with the system base only half changed. Accordingly, if the HG is not locked the condition code CC in one of the status registers is set to 2, (block 1508) which in this case is the code indicating the hardware gate HG is not locked.

Accordingly, the SWSB instruction is not allowed to proceed 1509, until the hardware gate HG is locked. If on the other hand the hardware gate HG is locked a determination is made as to whether or not the new system base word (i.e. address syllable AS) is quad-word aligned. If the AS is not quad-word aligned then the condition code CC is set to 3 indicating this condition and the SWSB instruction terminates 1509. On the other hand if the AS is quad-word

aligned the condition code CC 1512 is set to 0 which indicates that the SWSB instruction was successful and can proceed to perform the move operations 1513 from the new system base image into the system base area located at BAR. Finally after the new system base image has been transferred into the BAR system base the hardware gate is unlocked 1514, the vacant indicator in the auxiliary memory 1317a is set to 1, (block 1515) indicating no running process, and finally the instruction is complete 1516.

Figure 15c is a more detailed flow diagram of block 1513 which is the step that transfers the new system base image to the BAR system base area, and includes loading of the scratchpad memory S.P.M. with information from new system base image.

Referring to Figure 15c the start 1517 of SWSB instruction is actually at block 1513 of 15b. Typically 60 bytes of the new system base image located at address AS are transferred 1518 to the system base whose address is indicated by the boundary address register BAR. A new G table word GTW from the new system base image is loaded 1519 into scratchpad memory S.P.M. in auxiliary memory 1317a. Then a new J table word JTW is loaded 1520 in S.P.M. Finally a new internal processor queue word IPQW is loaded 1521 into S.P.M., and that portion of the instruction completes 1516 and continues to block 1514, where the operations are as previously described.

GLOSSARY OF TERMS

Absolute Address	- the physical address of a hardware byte in main memory store.
Address Development	- a hardware function which operates on a number of address elements to compute an absolute address which is used to refer to a location in main storage.
Addressing	- locating an object by any of a number of virtual, logical, and physical means.
Address Space	- the set of logical, segmented addresses corresponding to a process that the CPU is permitted to transform into absolute addresses during execution of that process.
Address Space Word	- one of two words in a process control block which point to the segment table word array. The segment table word array defines the segment tables associated with the process.
Address Syllable	- a logical address recognized by CPU hardware, normally an operand of an instruction.
Analyzer/Translator	- that part of the static linker that first passes the control language stream to the linker; then, if the control language is error-free, the analyzer/translator converts it to tables and other structures for the static linker to use.

Asynchronous Processing	- concurrent activation of more than one process.
Auxiliary Memory	- is comprised of 64 flip-flops for storing miscellaneous states of the system and is located in ALU.
Base Register	- the primary element in segment addressing, referenced by number in every address syllable.
Boundary Address Register	- a visible hardware register that defines the lowest memory address accessible to firmware/software.
Call	- see procedure call - Inward Call-Outward Call.
Central Process Unit	- a part of a computer that includes circuits controlling the interpretation and execution of instructions.
Channel	- a means of communicating between the processor subsystem and a peripheral subsystem. There are two types of channels, physical and logical. A physical channel is the hardware connection between the input/output control unit IOC and the peripheral control unit PCU. A logical channel is a software communications path between main memory and a single peripheral device.
Channel Command Entry	- an instruction in a channel program.
Channel Command Word	- an element of a channel command entry. Two channel command words constitute a channel command entry.

Channel Program - a sequence of instructions that cause a specific input/output i/o operation to be performed by a peripheral device.

Complementary Code - a code in some instructions which together with the operation code defines the function of the instruction.

Compilation Unit - the single object-code module that results from compiling or assembling one procedure in any high level language processor. The CU is not executable unit it is linked.

Concurrency - apparent simultaneity.

Condition Field - a 4-bit field in a branch instruction. Each bit of the field corresponds to a setting of the condition code in the status register bits 0 and 1.

Contest Indicator - located in auxiliary memory and indicates a new entry was placed in the ready queue.

Data Address Space - the set of logical record addresses accessible by a process through data management; consists of four elements; logical fields, logical records, data base pages, and files.

Data Descriptor - an information structure which is used in indirect addressing and as a description in data manipulation.

Decor - the functional characteristics or architecture of a computer system.

Decor Extension Mode - a mode of operation permitting the operating of the system in emulation mode.

Descriptor	- a 32- or 64-bit field used in developing the address of any data field. It is one of many pointers in the addressing scheme of the invention.
Descriptor Block	- a data block containing any number and variety of descriptor declarations except semaphore descriptors. No other data is allowed.
Device Adapter	- electronics which adapts a specific device to a peripheral controller.
Device Adapter Interface	- the interface between a device adapter and its associated peripheral controller(s).
Device Control	- a software facility which handles device specific characteristics.
Dispatcher	- the firmware responsible for dispatching processes in the CPU.
Dispatching	- a field within the instruction format or a data descriptor which provides relataive position information within a segment.
Displacement Addressing	- a form of direct addressing in which the operand defines a location relative either to the base of the immediate segment or the current ation counter.
Dynamic Addressing	- addressing which results in resolution, during execution, of references defined within the process group.
Editor	- that part of the static linker that format and outputs any or all of the information that results from linking a particular procedure.

Effective Address	- a logical, as opposed to physical, address consisting of a segment table number, a segment table entry, and a segment relative address. These elements show the way to a particular segment descriptor, which eventually points to the segment being referenced.
Emulation Mode	- operation of a computer so as to initiate another computer, and appear to operate in the other computer's environment.
Exception	- an exception occurs when the hardware detects the existence of some condition that has been specified as requiring special handling.
Event	- anything observed in a system by a process which may be of interest to some other process.
Firmware	- that part of a hardware unit implementation which utilizes a form of micro-programmed control.
G, D Name	- the name used by the hardware to refer to a semaphore. G, the G-segment name is the number of the entry in the G-table containing the segment descriptor; D is the relative address in the segment.
G-Segment	- a segment containing semaphores which may be addressed using the G, D name form and associated mapping facility (the G-table).

G-Table	- a table used for storage mapping (locating a G-segment) when the G, D name form is used.
Gating	- a facility for controlling entry to procedures in a (gated) segment through the use of procedure descriptors.
General Register	- a 32-bit register available to executing processes which generally holds binary or bit string data. Certain general registers can be used for indexing (GR8 through GR15).
Index	- a data structure which is maintained and operated upon by the system. The structure is visible to the user only in terms of a key which he presents for the purpose of making available or positioning to a specific record occurrence.
Indexing	- modification of an address by a signed arithmetic value by adding something to something.
Index Register	- a general register used for indexing (GR8 through GR15).
Indirection	- a method of addressing whereby an address is used to obtain another address, rather than the actual data.
Indirection to Base	- the form of indirection whereby a data descriptor containing a base register number and a displacement is found at the address referenced.
Indirection to Segment	- the form of indirection whereby a data descriptor containing a segment address is found at the address referenced.

Indirect Segment Descriptor	- a type of segment descriptor which contains information for addressing another segment descriptor, rather than a segment.
Instruction	- the unit of execution of a central processor visible to a programmer.
Instruction Counter	- a register associated with an executing process which contains the segmented address of the next instruction of the procedure being executed.
Interleave	- sequential accessing of memory modules to reduce memory access time.
Interrupt	- the discontinuance of a process in the system due to an event occurrence.
Interrupt Process	- a process which is activated in response to an interrupt.
Inward Call	- when one procedure execution in one ring number calls another procedure to execute in a lower ring number.
I/O Controller	- a unit which provides the basic control for a specific I/O subsystem.
I/O Processor	- potentially asynchronous system processes which deal primarily with the movement of data between peripheral storage or I/O devices and main storage.
J.P. Tables	- a collection of logical addresses for locating a process control block.
Job	- a unit of work for the system.
Job Control Language	- a language for describing a job and its component parts.

Job Step	- the major subdivision of a job leading to the execution of a program defined by a JCL statement.
Job Step Table	- a system table used to map the first part of a process name, i.e., to locate the process group table.
Linked Module	- the output of the static linker. It is a consolidated group of compilation units (CU) whose cross-references were resolved by the static linker.
Linking	- see static linking.
Logical Channel	- see channel.
Logical Processor	- the collection of hardware resources and control information necessary for the execution of a process.
Main Storage	- all addressable storage from which instructions can be executed or from which data can be loaded directly into registers.
Magnetic Tape Controller	- the elements of the peripheral subsystem related to magnetic tape devices.
Mass Storage Controller	- the elements of the peripheral subsystem related to mass storage devices.
Memory Management	- operating system facilities for the allocation, relocation, and disallocation of physical memory.
Message	- information transmitted from a source or to a destination—neither being a file.
Microinstruction	- same as microinstruction word, and control store word.

Micro-program	- a set of machine codes used to implement the control functions of a processor.
Multiplex	- to share a hard resource, such as memory, usually by time division.
Multi-programming	- the concurrent execution of two or more programs by a single computer.
Native Mode	- operation of a computer in its native environment.
Offset	- in address development, the number of bytes after the beginning of the segment at which an addressed part of the segment starts.
Operating System	- a system of basic software designed to support the efficient operation of user software.
Outward Call	- when one procedure executing in one ring number calls another procedure to execute in a higher ring number.
Peripheral Control Unit	- a stand-alone micro-programmed processor which executes channel programs to perform I/O operations.
Peripheral Subsystem Interface	- a standard interface used for transfer and control between free-standing peripheral control units and I/O control units.
Physical Channel	- see channel.
Physical Data Structure	- a complete definition of the organization of data as it is physically recorded on media.

Physical I/O - that part of the operating system which initiates and controls the transfer of data between memory and peripheral or terminal devices.

Physical Storage - hardware used for the storage of data. It is composed of various types of recording media and the recording/reading hardware.

P-operation - a machine instruction involving a semaphore which causes a process to enter the wait state or to receive a message.

Privileged Instruction - an instruction which must be executed in ring 0.

Procedure - a named software function or algorithm which is executable by a processor without concurrency: a Fortran subroutine, a Cobol program, a PL/1 internal procedure.

Procedure Call - a mechanism for creating a stack frame within a stack segment.

Procedure Descriptor - a word containing the location and entry point of a procedure.

Procedure Segment - the type of segment whose contents represents a procedure, procedures, or part of a procedure.

Process - the ordered execution of instructions without concurrency by a processor, either central or i/o.

Process Address Space - the memory addresses which may be referenced or to which control may be transferred during a particular process.

Process Control Block	- a data structure (hardware defined and recognizable) that contains information needed to specify the state of a process at any given time.
Process Group	- a related set of processes, usually those necessary for the performance of a single job-step.
Process Group Builder	- that part of the static linker whose actions yield the linked module.
Process Group Loader	- an operating system facility that makes final resolutions in a linked module to create an executable process group then it loads the process group into memory and starts its execution.
Process Group Table	- see P Table.
Process Link	- an entry in a ready process or waiting process queue.
Process State	- the dynamic condition of a CPU process e.g., running, ready, waiting, or suspended.
Process Switch	- the hardware function that disassociates one process from the central processor and associates another.
Process Synchronization	- the functions that act to coordinate the asynchronous activity among processes. These functions generally involve semaphores and the P and V operations.
Processor	- a unit capable of receiving data, manipulating it, supplying results, and controlling its sequence of operations in response to stored

programs: short for central processor, generic for central processor, peripheral processor, or hardware/ software processor.

Program - the specification of the procedural and associated information necessary to solve a problem, i.e. the ordered collection of instructions which is executed by the computer and operates on data to perform a user's job or some phase of that job.

Program Execution - the activity of a process in conformance to program specification.

Protection - the function, enforced by hardware and software, of preventing processes from interferring with each other or sharing each other's address space in an unauthorized manner.

P-Table - a hardware defined data structure containing entries, each pointing to the origin of a process control block. The P-table entries point to the total set of PCB's comprising a process group at any given time. Also called process group table.

Q/PR/RDY - queue of processes ready i.e. a ready-queue.

Queue - an ordered list of entities awaiting information, process or availability.

Ready State - a process state in which no processor is associated but all necessary resources, except a processor, are available for changing to the running state.

Relative Address	- the incremental/decremental location of some object relative to another.
Relocation	- movement of a segment from one location in main storage to another including the adjustment of all required references to its previous location.
Return	- the function and operations necessary to resume the action of a procedure at a point immediately following the point where it entered some other procedure.
Ring	- a protection attribute of a segment that restricts the read, write, and execute access of process to that segment. A level of privilege of a process for read, write or execute.
Rolling-in	- performing the operation necessary to give control of a processor to a new process.
Rolling-out	- performing the operations necessary to take away control of a processor by a process.
Running State	- the dynamic condition of a process (connected to a CPU) that is executing.
Scratch Pad Memory	- 256 locations in ALU for storing CPU control information-also known as a local store unit.
Scheduling	- determining the sequence of operations.
Scientific Registers	- a 64-bit register used in the manipulation of binary floating

point numbers. There are four scientific registers in the invention when the system has the scientific option.

Segment - a contiguous main memory space treated as a unit.

Segmentation - the use of a division of main memory into logical groupings called segments rather than into a single linear store.

Segment Base - the current origin of the segment. This is one of the fields in a segment descriptor word.

Segment Descriptor - entry in a segment table defining the characteristics of a segment or pointing to a segment descriptor.

Segmented Address - see effective address.

Segment Number - the identification of a particular segment consisting of a segment table number (selecting one of the segment tables of the process) and a segment table entry of that chosen table.

Segment Relative Address - during address development, the final value to be added to the segment base to obtain the absolute address.

Segment Table - a table containing segment descriptors used to establish the addressability of a process to its segments. The collection of segment tables of a process defines the address space for that process.

Segment Table Entry	- an item in a segment table. Each segment table entry is a segment descriptor.
Segment Table Word	- an entry in a segment table word array pointing to the segment tables of a process. The array is pointed to by one of two address space words in the process control block.
Semaphore	- data structures used to control communication between processes.
Semaphore Block	- a data block containing only semaphore descriptor declarations.
Stack	- a mechanism that accepts stores, and allows retrieval of data on a last-in-first-out basis. Comprised of a number of contiguous portions called stack frames.
Stack Active Area	- the portion of the current stack frame that holds a pointer to the saving area, the caller's instruction counter, parameter space and local variables—in short, the data of the current frame most likely to be useful to execution of the called procedure.
Stack Base Words	- a group of three words in a process control block which contain the segmented address of the stack segments for rings 0, 1, and 2 associated with the process.
Stack Frame	- a contiguous part of the stack segment holding the data stored for a single procedure call.

Stack Operations	- the push-down and pop-up operations of storing and retrieving information to/from the stack. In practice, manipulation of the pointers used to reference the information stored in the stack.
Stack Overflow	- the action of attempting to store more information into the stack then the available space can hold. This is detected by hardware.
Stack Register	- see T register.
Stack Segment	- the segment of a process that contains its stack.
Static Linking	- an intermediate step between compiling a source procedure and executing the object form of that procedure; linking resolves the external references made to and from that procedure.
Status Register	- an 8-bit register which specifies the status of the procedure currently being executed.
Swapping	- releasing the main memory space occupied by a segment by writing the segment out to secondary storage. The actual write to secondary storage may be suppressed if the segment has not been modified since the last time it was read in.
Synchronization	- having a fixed relationship in time; synchronized execution of two or more processes.
System Base	- a fixed area in main memory which acts as a root for all information structures.

Task	- the smallest unit of user-defined work consisting only of a single stream of non-concurrent instructions.
T Register	- a software visible hardware register which contains the segmented address of the top of the procedure stack.
Unit Record Controller	- the elements of the peripheral subsystem related to unit record devices such as card equipment, paper tape equipment, and line printers.
User Process Group	- the internal representation of any job step, as opposed to a system process group which exists independently of any job.
Vacant Indicator	- a hardware structure that indicates that the process which had control of the CP has suspended itself i.e. no CJP in the CPU. Located in Auxiliary Register, of the ALU.
Virtual Memory	- an addressing concept that allows a programmer to code without regard to physical memory size; virtual memory manager automatically manages the exchange of segments between main memory and secondary storage.
V-Operation	- a machine instruction intended for interprocess signaling the completion of an event. A V-OP operates on a designated semaphore.

ABBREVIATIONS

ACU	- address control unit
ALU	- arithmetic and logic unit
AS	- address syllable
ASW	- address space word
BAR	- boundary address register
BCD	- binary coded decimal
BR	- base register
CCE	- channel command entry
CCU	- channel control unit
CMRN	- the callers maximum ring number
CCW	- channel command word
CET	- current state entry time
CIA	- control store interface adapter
CJP	- currently executing process
CPU	- central process unit
CU	- compilation unit
CSU	- control store unit
D	- displacement
DMU	- data management unit
DA	- device adapter
EAR	- effective address ring
EXW	- exception word
GR	- general register
GTW	- G-table word
IC	- instruction counter
IFU	- instruction fetch unit
I/O	- input/output
IOC	- input/output controller
IPQW	- internal process queue word
IR	- index register
ITBB	- indirect to base register
ITS	- indirect to segment

J	- job
JCL	- job control language
JTW	- J-table word
LCT	- logical channel table
LIFO	- last-in-first-out
LSU	- local store memory or scratch pad memory
MBZ	- must be zero
MOS	- metal oxide semiconductor
MAXR	- the maximum ring number at which a procedure may execute MAXR is found in the segment descriptor of SEG_{EP}.
MSC	- mass storage controller
MTC	- magnetic tape controller
NFS	- non-functional status
NPRN	- new process ring number
NJP	- new process replacing the currently executing process
PCB	- process control block
PCT	- physical channel table
PCU	- peripheral control unit
PL/I	- programming language/one
PMW	- process main word
Q/PR/RDY	- queue of processes ready
RD	- read ring
RHU	- reserved for hardware use
RPW	- running process word
RSU	- reserved for software use
RTA	- ready time accounting
RTO	- residual time out
SBW	- stack base word
SEG	- segment number
SKW	- stack word
SEG_{EP}	- the segment which contains the entry point as found in the procedure descriptor

SEG_{PD}	- the segment which contains the procedure descriptor
SR	- scientific register
SRA	- segment relative address
STE	- segment table entry
STN	- segment table number
STR	- status register
STW	- segment table word
STWA	- segment table word array
T	- T-register (stack register)
URC	- unit record controller
WR	- write ring
WTA	- waiting time accounting

Having shown and described one embodiment of the invention, those skilled in the art will realize that many variations and modifications can be made to produce the described invention and still be within the spirit and scope of the claimed invention.

What is claimed is:

* * *

For Claims 3, 5 and 6, see Rejected Claims 3, 5, and 6.

DECLARATION AND POWER OF ATTORNEY (use with sole or joint patent applications accompanying or not accompanying the application papers)

(X) the undersigned declare that the information set forth above and all facts stated herein are true; that (X) he believes that (X) he is the original first and (X) joint inventor of the invention described and claimed in the specification (X) attached hereto or (__) of application Serial Number _____ filed _____; that he has reviewed and understands the contents of the above-identified specification, including the claims, as amended by any amendment referred to above; that he does not know and does not believe that the same was ever known or used in the United States before (X) his invention thereof or patented or described in any printed publication in any country before (X) his invention thereof or more than one year prior to such application; that said invention has not been in public use or on sale in the United States for more than one year prior to the date of this application; that said invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States on an application filed by (X) him or (X) his legal representatives or assigns more than twelve months prior to the date of this application; that he does not know and does not believe that the same was ever known or used in the United States before (X) his invention thereof or patented or described in any printed publication in any country before (X) his invention thereof or more than one year prior to the date of this application or in public use or on sale in the United States more than one year prior to the date of this application and that said invention has not been patented in any country foreign to the United States on an application filed by (X) him or (X) his legal representatives or assigns more than twelve months prior to the date of this application; that he does not know and does not believe that any person, firm or corporation has any right, title or interest in the invention except as described in this declaration; that no application for patent or inventor's certificate on said invention has been filed by (X) him or (X) his legal representatives or assigns in any country foreign to the United States except as follows:

20	Bradley	John	J.
0	Garches	France	U.S.A.
1	63 rue du 19 Janvier 92380	Garches	France
20	Franklin	Benjamin	S.
0	Cambridge	Massachusetts	U.S.A.
2	11 Peabody Terrace Apt. #2001	Cambridge	Massachusetts 02138
20			
0			
3			

(X) Further declared that all statements made herein of (X) his own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

SIGNATURE(S) OF APPLICANT(S) (X) HEREIN *	DATE OF SIGNATURE
John J. Bradley	8 April 75

301 TITLE OF INVENTION: Switch System Base Mechanism

600 CORRESPONDENCE ADDRESS

( ) Specific reference to any prior pending application(s) in which this application is a continuation, continuation-in-part or division of, and claiming benefit under 35 USC 120

(X) Hereby appoint the following attorney(s) to prosecute this application and transact all business in the Patent Office connected therewith

NAME	REGISTRATION NUMBER	ASSOCIATION NUMBER
Nicholas Prasinos Ronald T. Reiling	24,791 20,607	(X)

Send Correspondence to: Ronald T. Reiling, Honeywell Information Systems Inc., 200 Smith Street, Waltham, Massachusetts 02154, (617) 890-8400, Ext. 22126

(X) Further declares that all statements made herein of (X) his own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

SIGNATURE(S) OF APPLICANT(S) (X) HEREIN * | DATE OF SIGNATURE

John J. Bradley | 8 April 75

(X) * See attached sheet

WAS ORIGINALLY FI...

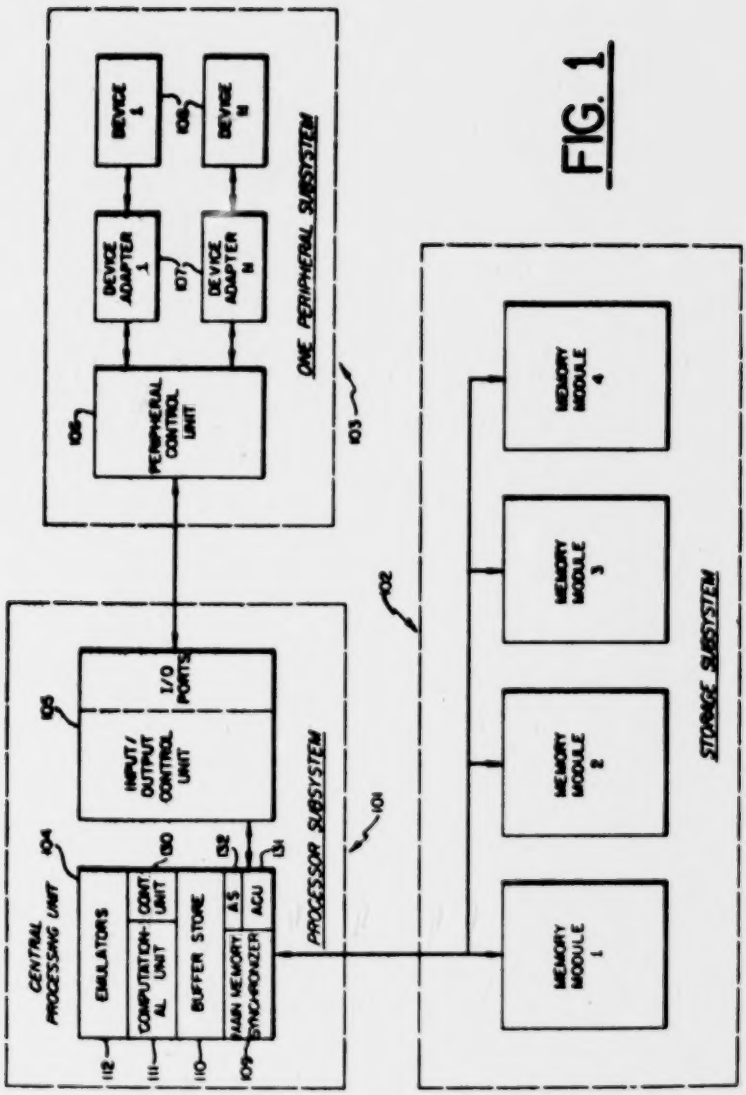


FIG. 1

570331

POINT OF DRAWING AS ORIGINALLY FILED

FIG. 2

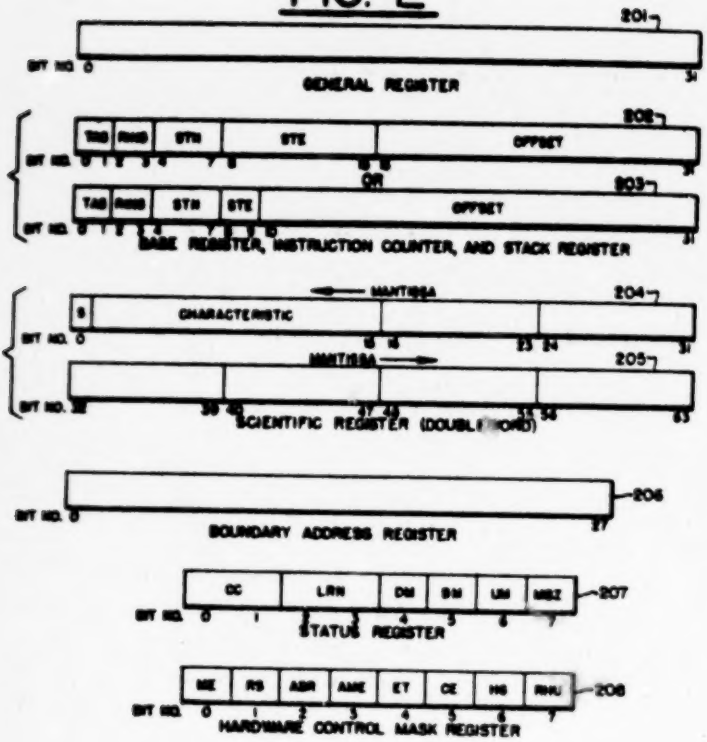


FIG. 3

LEGEND FOR RESERVED AREAS OF STORAGE IN REGISTERS

AME • ACCOUNTING MECHANISM
ABR • AUTO STORAGE RECONFIGURATION
BM • BINARY MASK
CC • CONDITION CODE
CE • CORRECTED ERROR
DM • DECIMAL MASK
ET • ENVIRONMENT TOLERANCE
MB • HARDWARE GATE
LRN • LAST RING NUMBER
MBZ • MUST BE ZERO
ME • MACHINE ERROR
RING • RING NUMBER
RMU • RESERVED FOR HARDWARE USE
RS • RETRY SUCCESSFUL
S • SIGN
STE • SEGMENT TABLE ENTRY
STN • SEGMENT TABLE NUMBER
TAG • DESCRIPTOR TYPE
UM • UNDERFLOW MASK

AS ORIGINALLY FILED

570331

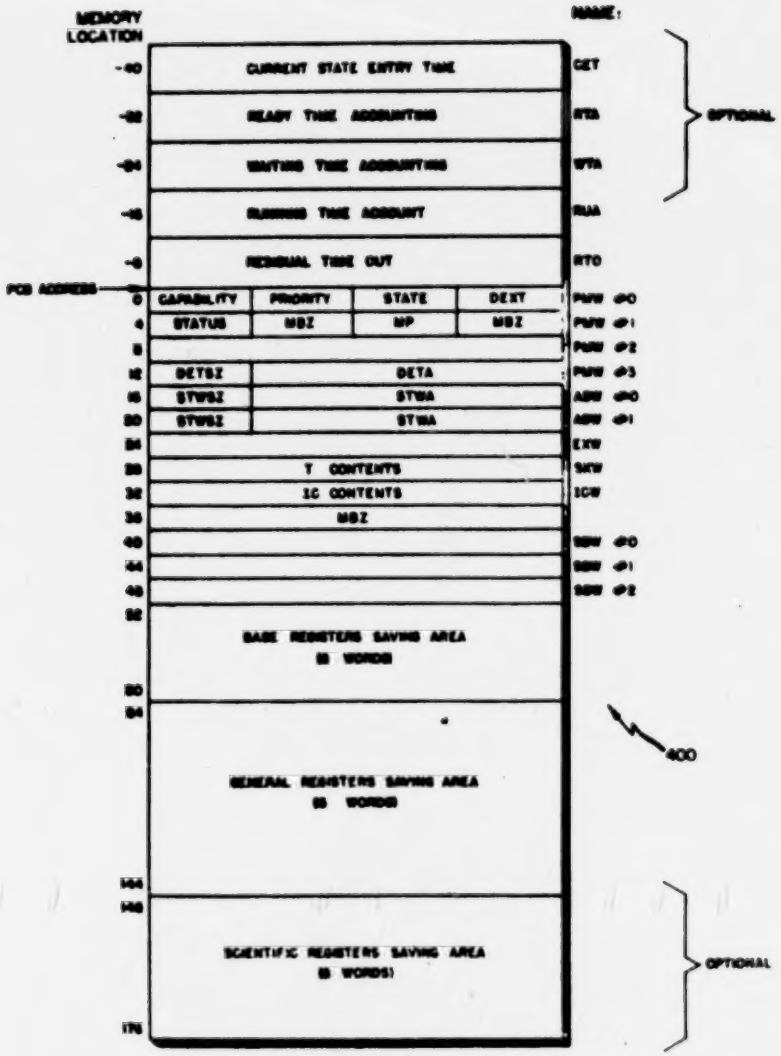


FIG. 4

SIZE OF DRAWING
HAS BEEN ORIGINALLY PLACED

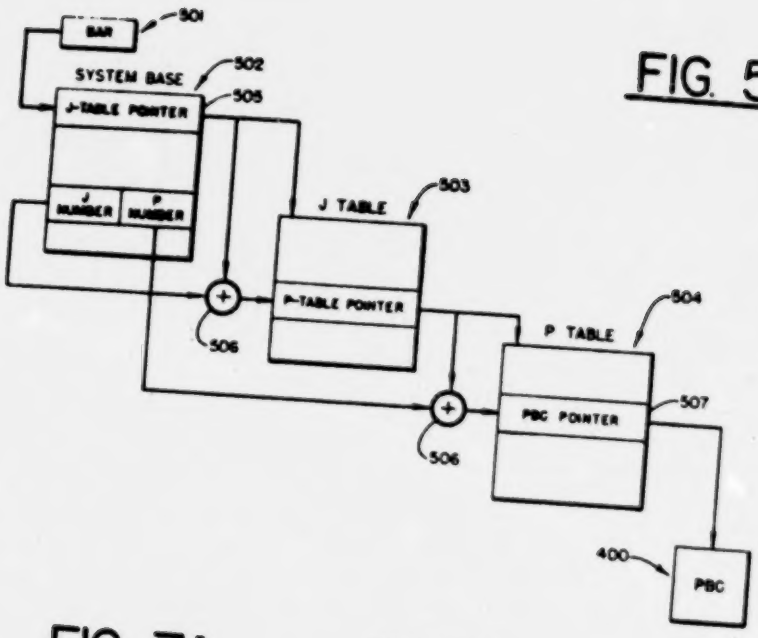
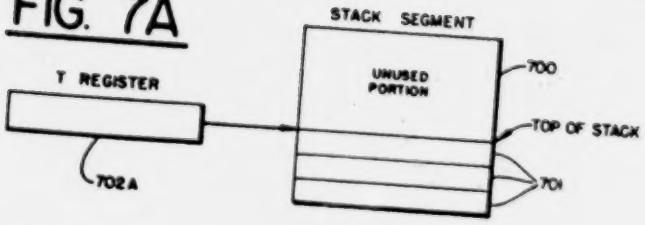


FIG. 5

FIG. 7A



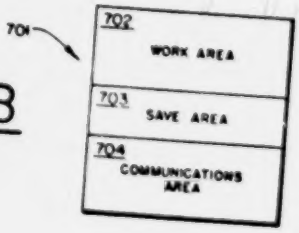
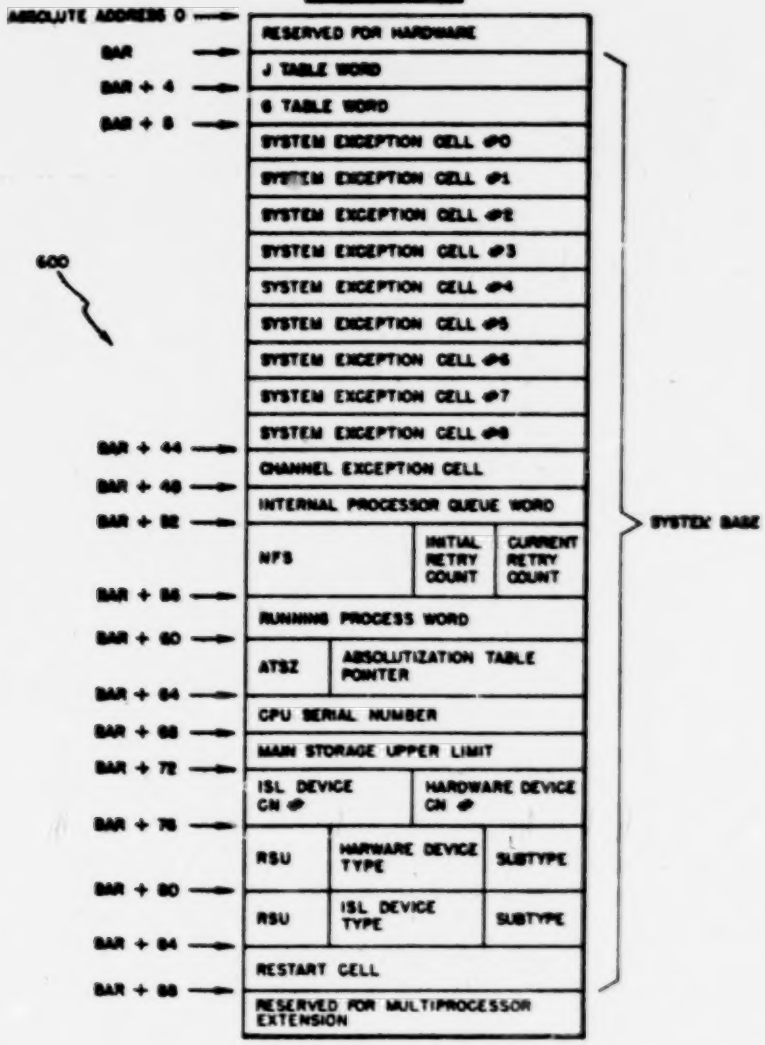


FIG. 7B

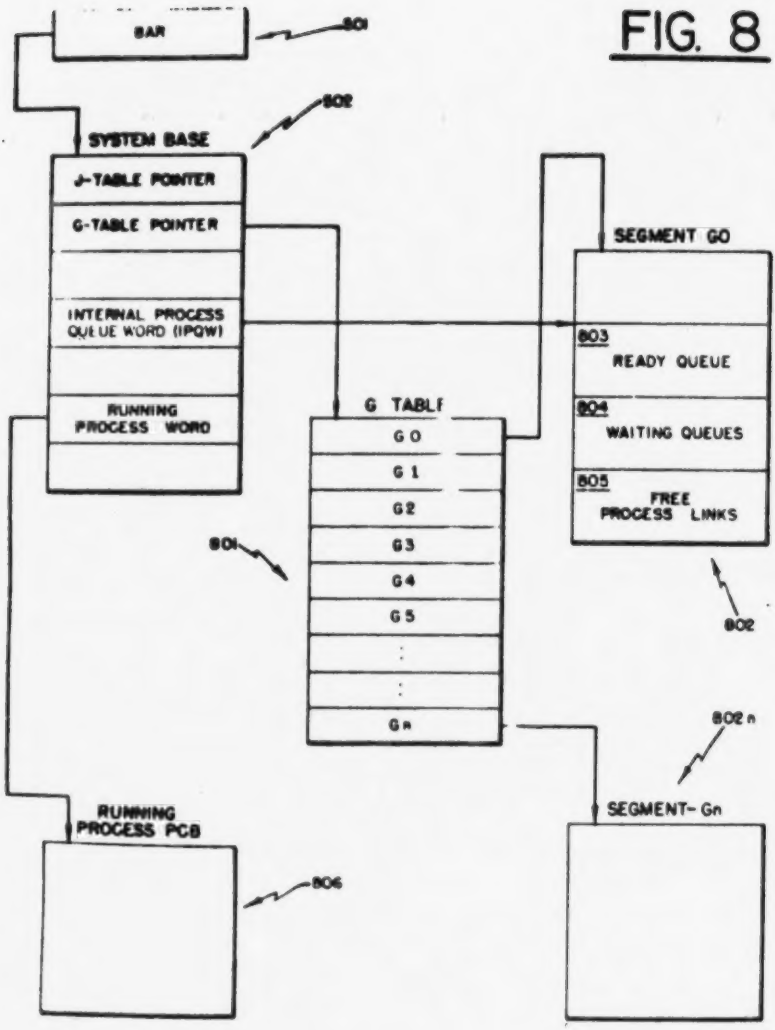
AS ORIGINALLY FILED

FIG. 6



.46 ORIGINALITY

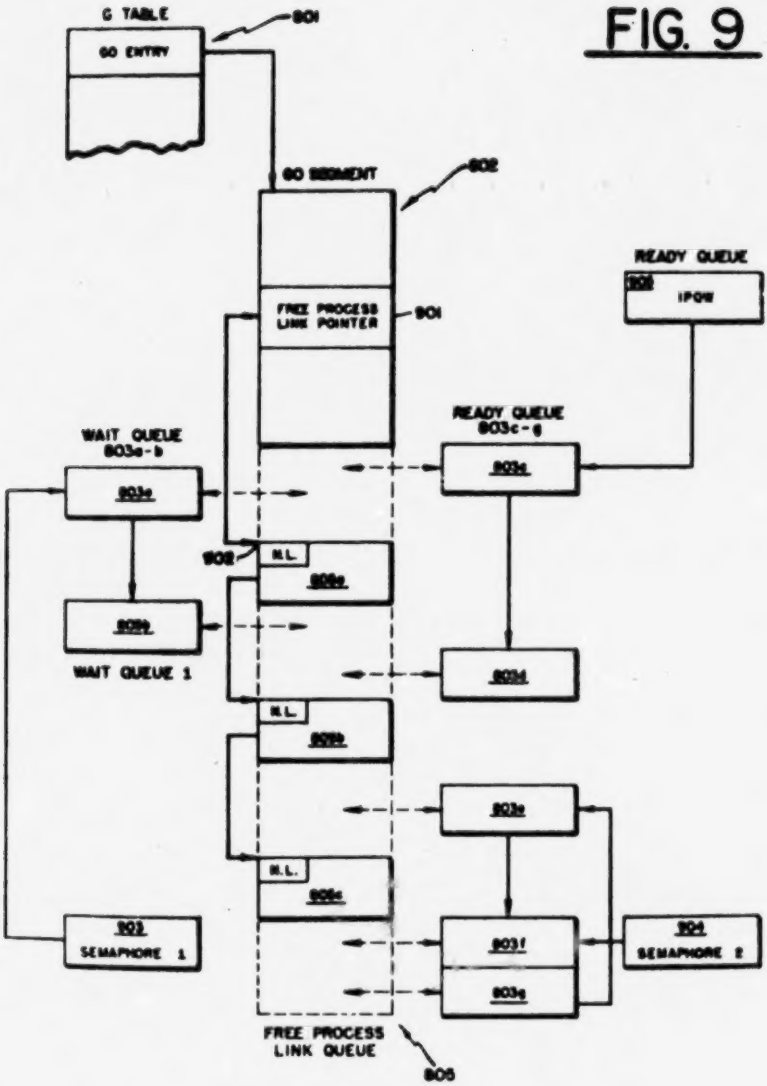
FIG. 8



AS ORIGINALLY FILED

570371

FIG. 9



AS ORIGINALLY FILED

570331

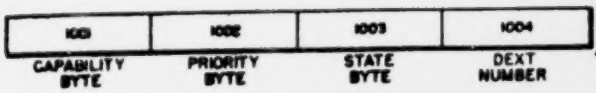


FIG. 10a

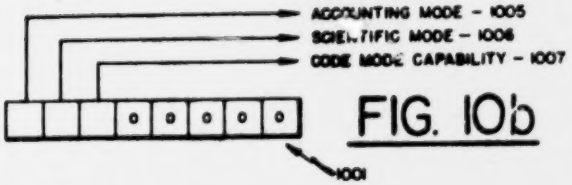


FIG. 10b

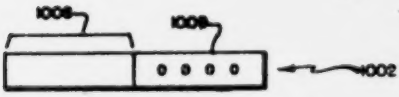


FIG. 10c

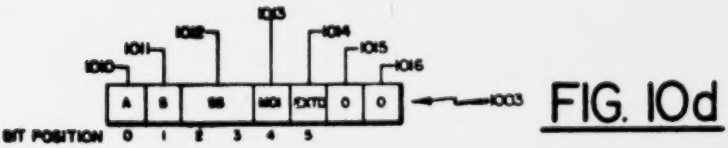


FIG. 10d

FIG. 10e

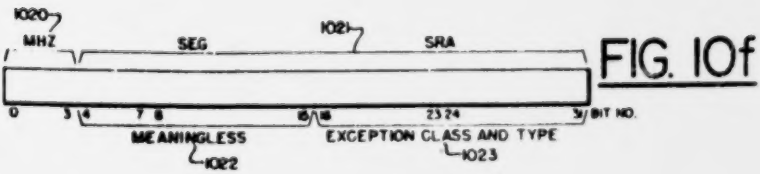


FIG. 10f

49 ORIGINALLY ...
57[illegible]11

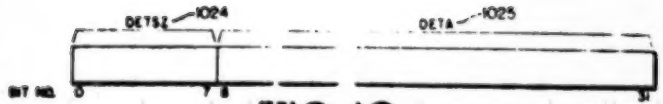


FIG. 10g

FIG. 10h

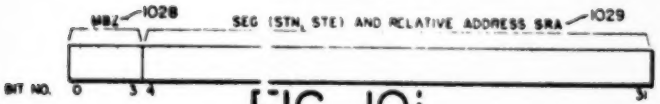


FIG. 10i

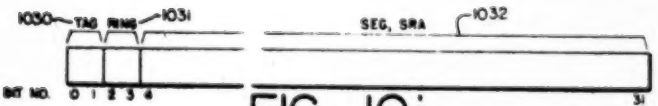


FIG. 10j

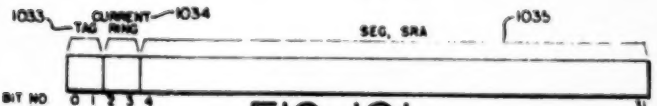


FIG. 10k

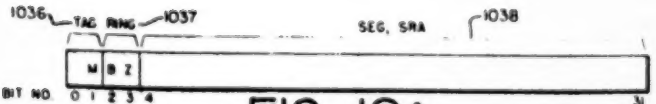


FIG. 10l

AS ORIGINALLY

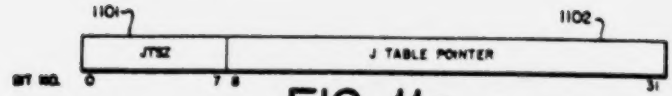


FIG. 11a

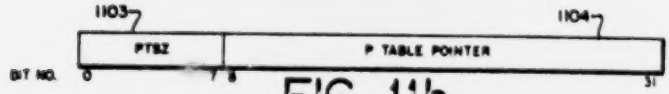


FIG. 11b

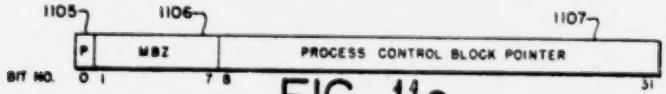


FIG. 11c

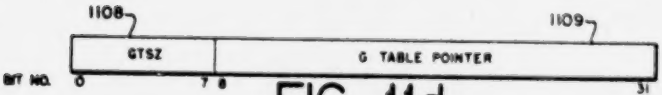


FIG. 11d

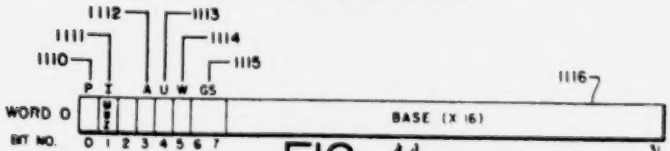


FIG. 11e

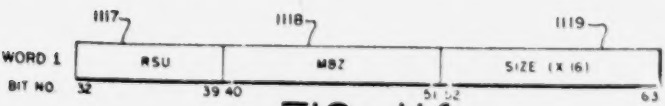


FIG. 11f

...AS ORIGINALLY FI...

570371

FIG. 11g

FIG. 11h

FIG. 11i

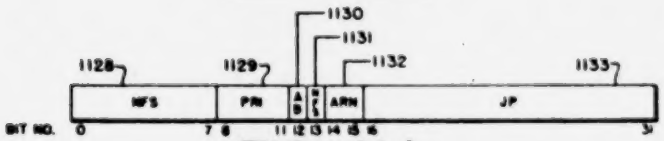


FIG. 11j

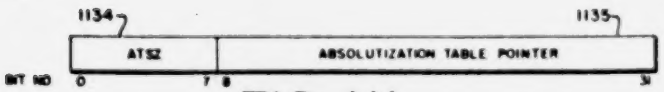


FIG. 11k

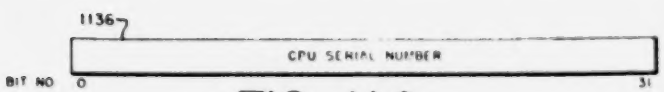


FIG. 11l

PRINT OF DRAWING
AS ORIGINALLY FILED

57:371

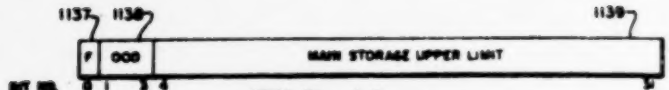


FIG. 11m

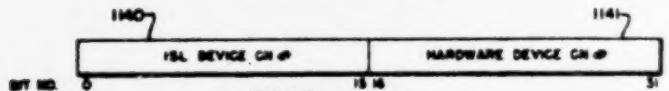


FIG. 11n

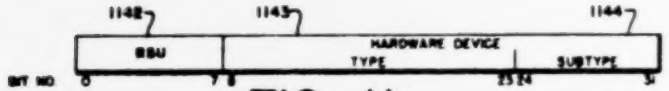


FIG. 11o

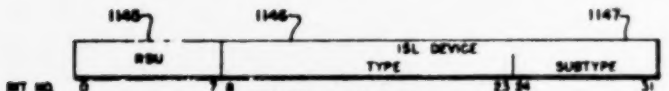


FIG. 11p

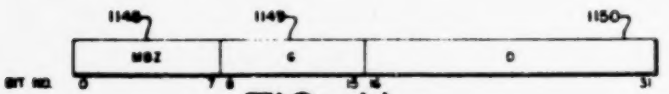


FIG. 11q

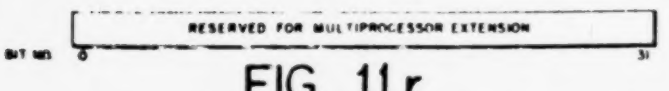
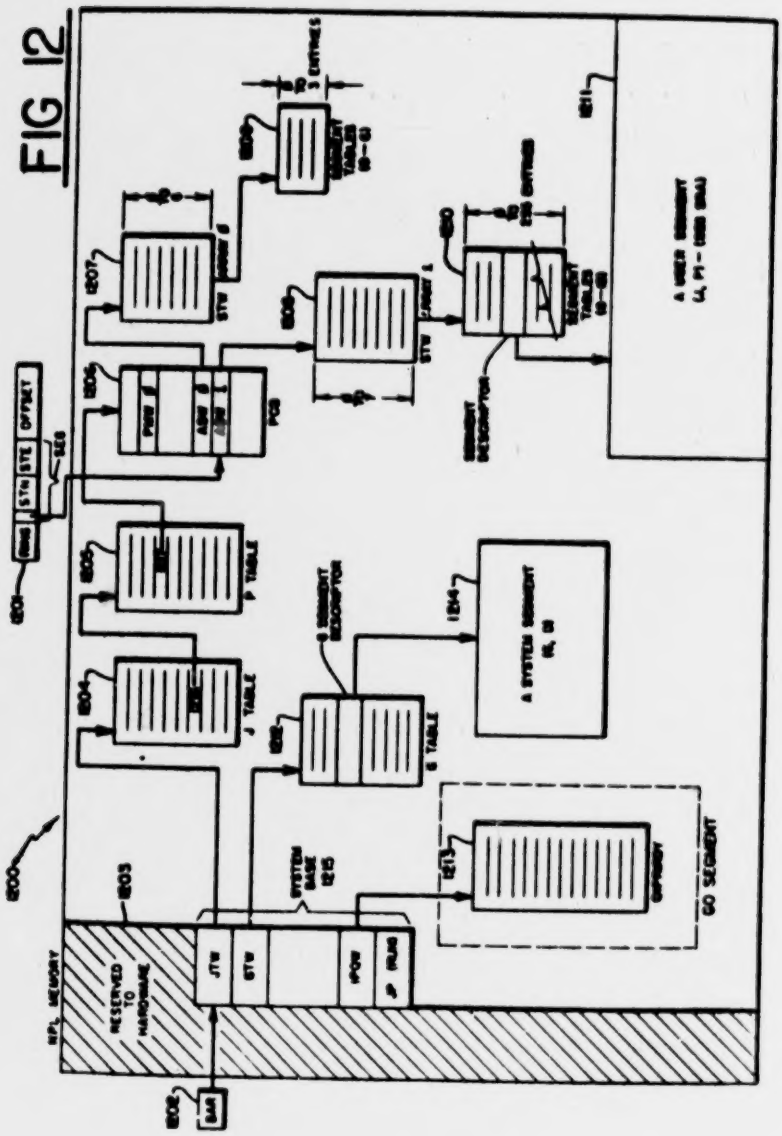


FIG. 11r

FILED WITH DRAWINGS
AS ORIGINALLY FILED

57C3'11

FIG 12



PRINT OF DRAWING AS ORIGINALLY FILED

570371

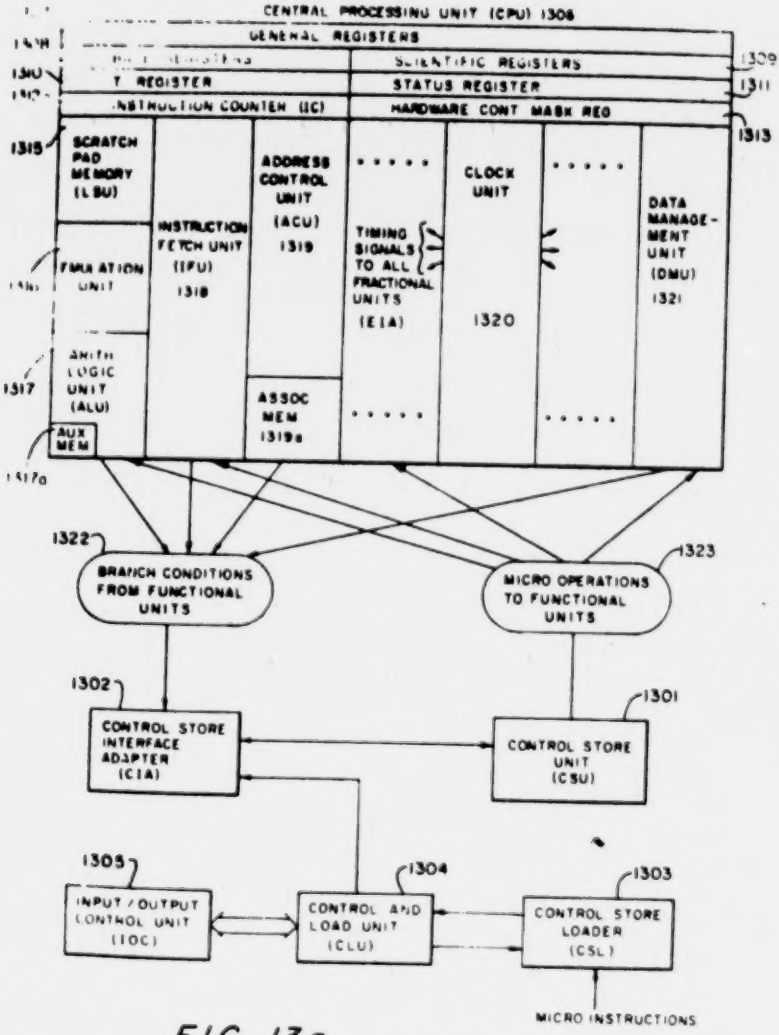
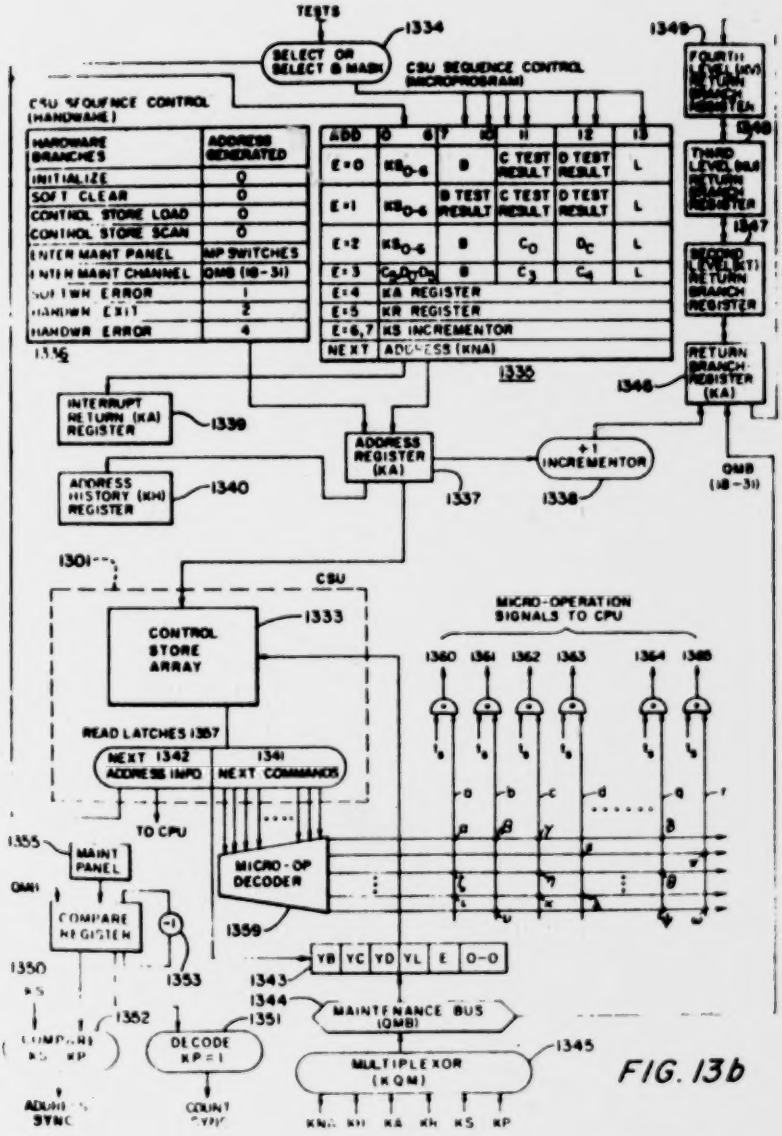


FIG. 13a

PART OF DRAWING AS ORIGINALLY FILED

07C331



570331

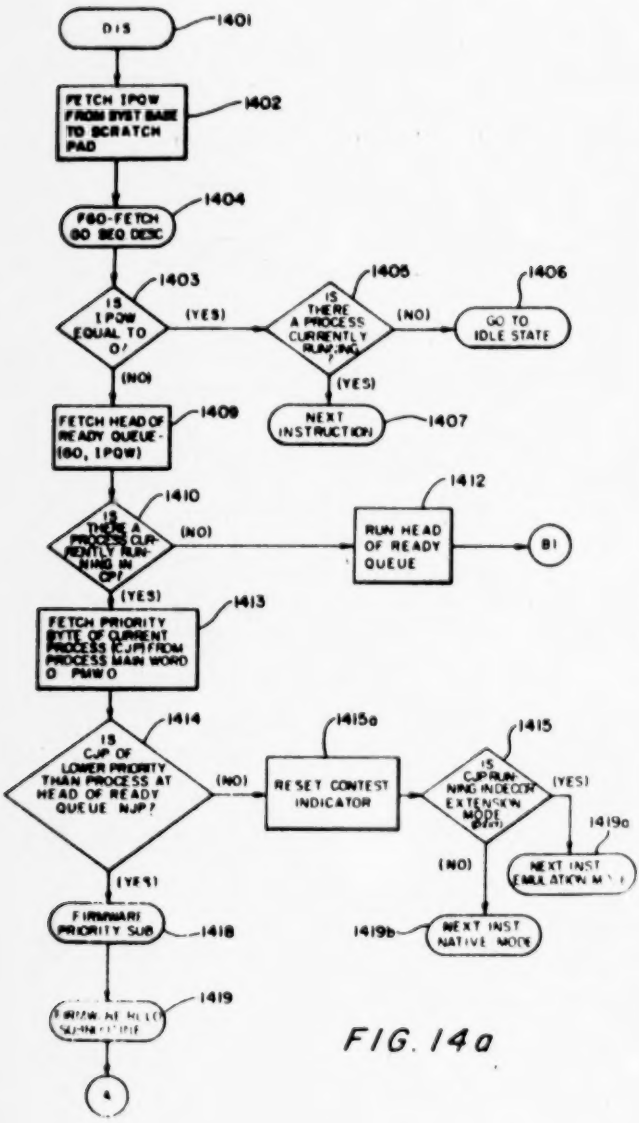


FIG. 14a

PART OF DRAWING
WAS ORIGINALLY FILED

57C331

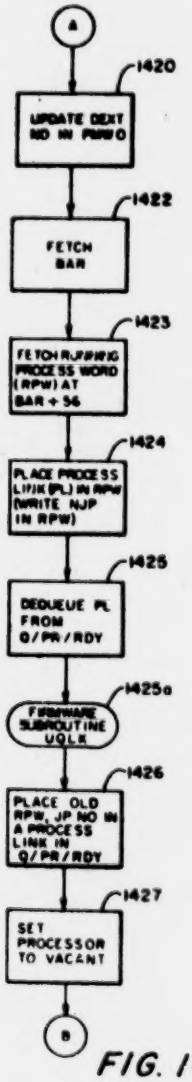


FIG. 14b

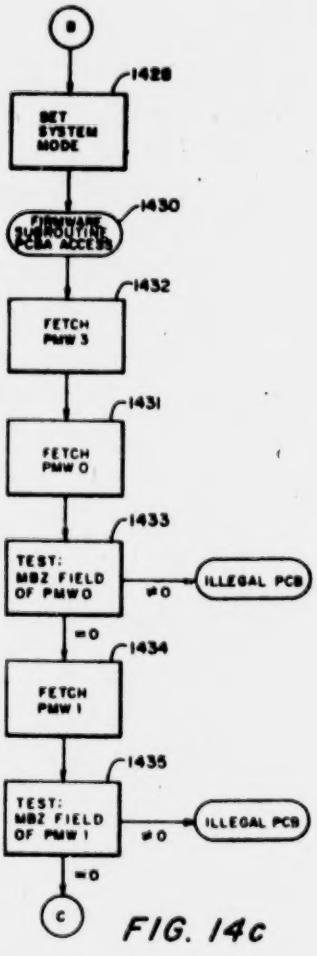


FIG. 14c

PRINT OF DRAWING AS ORIGINALLY FILED

57C371

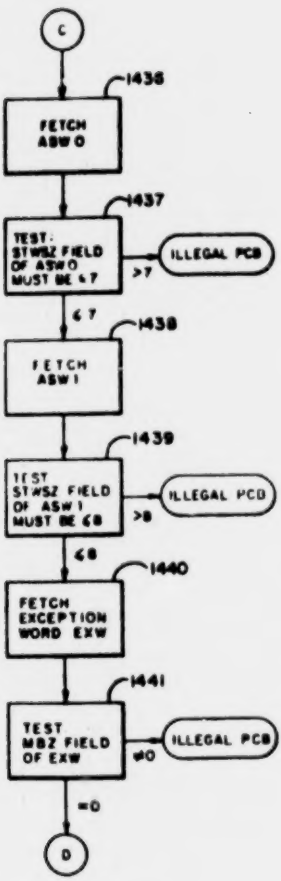


FIG. 14d

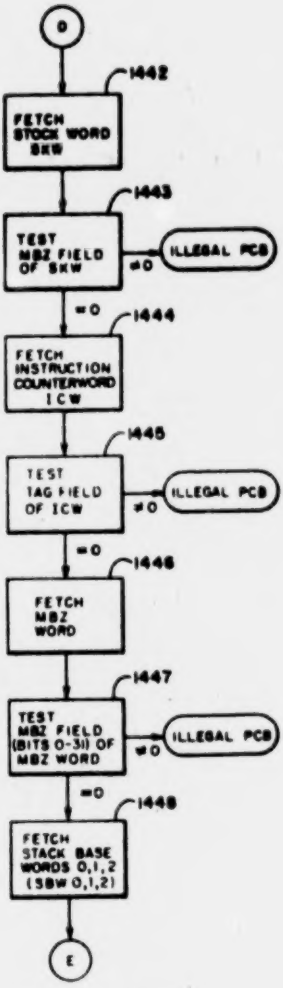


FIG. 14e

TITLE OF DRAWING
WAS ORIGINALLY FILED

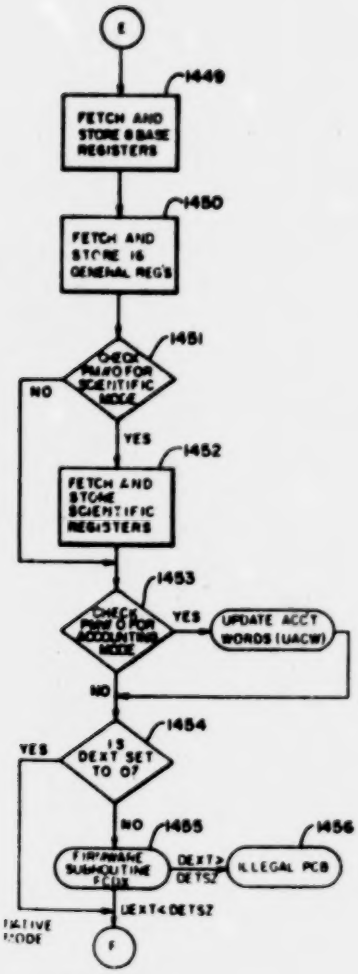


FIG. 14f

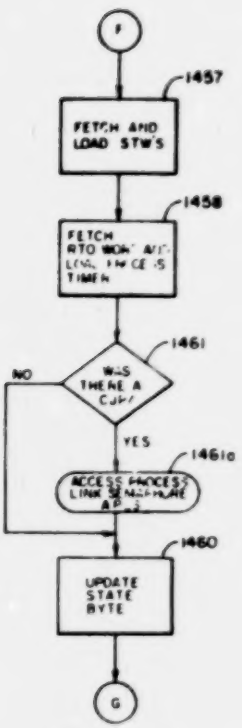


FIG. 14g

WAS ORIGINALLY FILED

570371

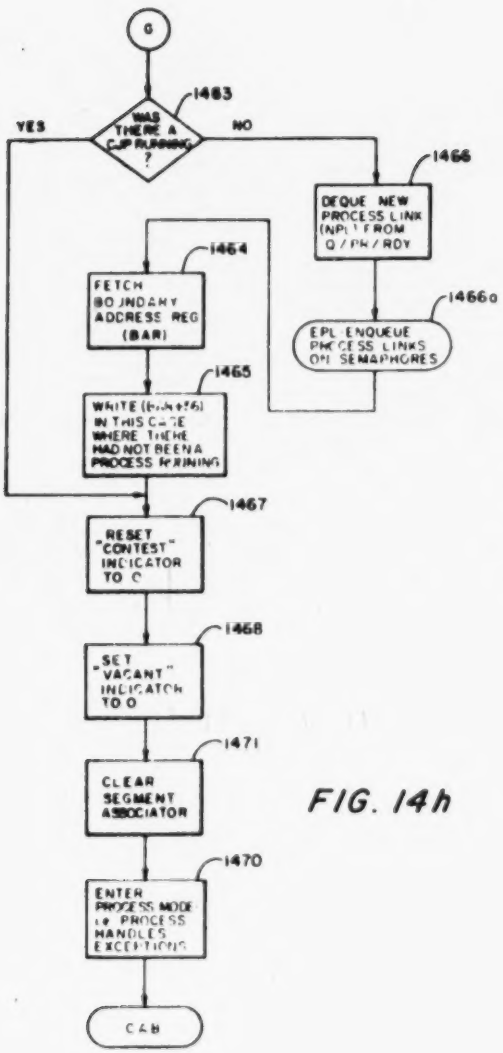


FIG. 14h

[illegible] OF DRAWING
WAS ORIGINALLY FILED

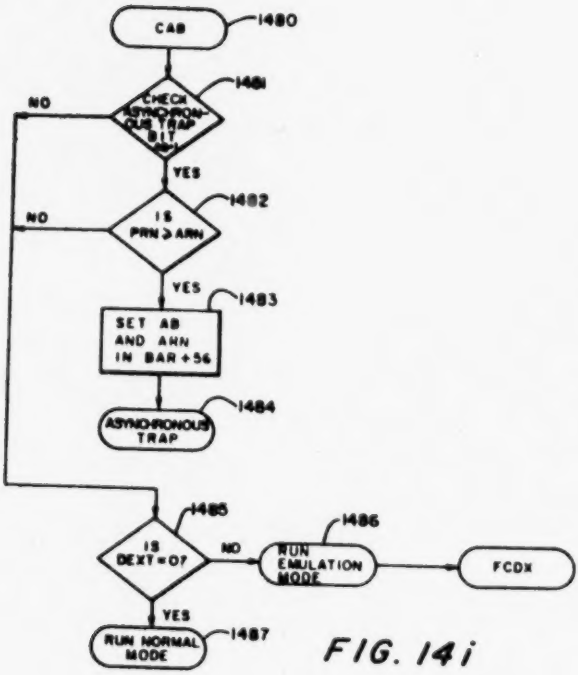


FIG. 14i

MAIN FIELD USES	SEQUENCE TYPE	BRANCHING AND / OR MICRO-OPS					CONSTANT GENERATION AND DESIGNATION	DATA TO BUS		MICRO-OPS	CHECKING
FIELD NAME	E	A	B	C	D	L	K	QA	QB	F	P
BITS	3	6	4	6	6	1		4	4		
		23					14	8		32	4

1326 1327 1328 1329 1330 1331 1325

FIG. 13c

570331

PRINT OF DRAWING
AS ORIGINALLY FILED

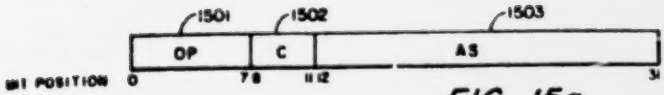


FIG. 15a

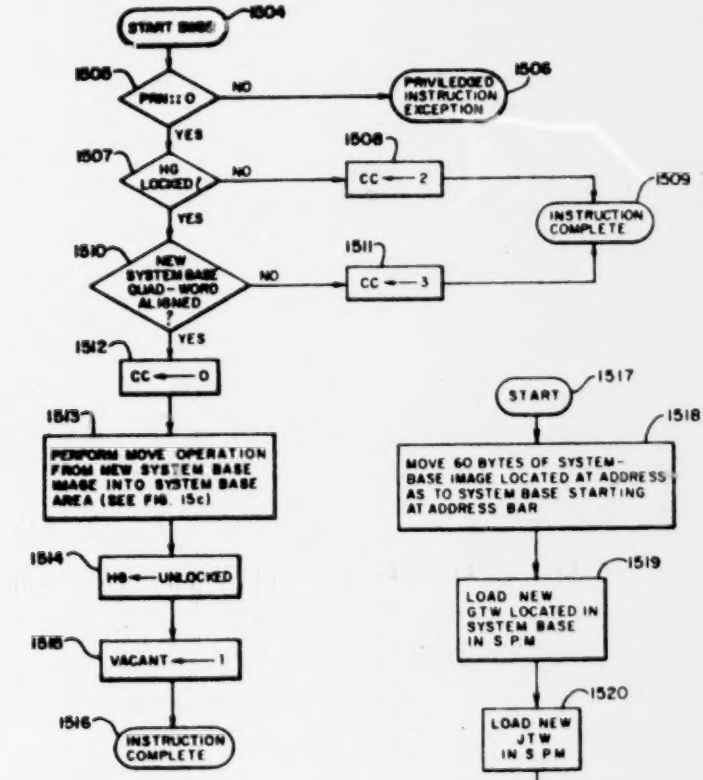


FIG. 15b

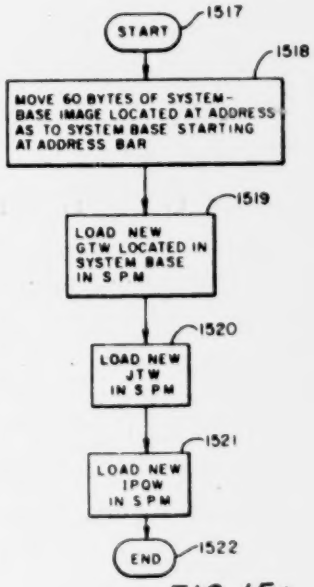


FIG. 15c

REJECTED CLAIMS.

1. In a multiprogramming computer system having a main memory, a central processing unit (CPU) coupled to said main memory, said (CPU) controlling the state of a plurality of groups of processes being in a running, ready, wait or suspended state, said computer system also having scratchpad registers being accessible to an operating system for controlling said multiprogramming computer system, a data structure for storing coded signals for communicating between said processes and said operating system, and said scratchpad registers, said data structure comprising:

(a) first means in said data structure and communicating with said operating system for storing coded signals indicative of an address for a selected one of said processes;

(b) second means in said first means for storing coded signals indicating priority of said selected one of said processes in relation to others of said processes for obtaining control of said CPU when ready;

(c) third means in said data structure and communicating with said operating system, for storing coded signals indicative of an address for a selected one of said plurality of groups of processes, and,

(d) fourth means coupled to said data structure and said scratchpad registers, for generating signals causing the changing of information in said date structure and said scratchpad registers.

2. In combination with a communication center located in a predetermined area of a computer main memory the computer having a central processing unit (CPU) for executing plurality of processes that compete with each other for control of said CPU, operating system means coupled for providing information to said communication center, and a scratch pad memory in said CPU which stores, under control of said operating system means, portions of information

located in said communication center, an improved system controlled by one of said processes for changing information in said communication area and in said scratchpad memory, said improvement comprising:

(a) first means for generating signals causing the creation of an image of said communication center at another location in said main memory;

(b) second means responsive to said operating system means for generating signals causing a change of the information in said image, resulting in a new image; and,

(c) third means, coupled for being responsive to said first and second means and to one of said processes for generating signals for causing a transfer of the new image to said communication center and to said scratchpad memory.

3. The system as recited in Claim 2 wherein said means means is a switch system base instruction in firmware.

4. The system as recited in Claim 3 further including an arithmetic and logic unit having an auxiliary memory said arithmetic and logic unit coupled for communicating with said first, second and third means.

5. The system as recited in Claim 4 wherein said auxiliary memory includes a hardware gate which has a locked and unlocked state said unlocked state permitting internal interrupts or external exceptions, and said locked state not permitting internal interrupts or extral exceptions.

6. The system as recited in Claim 5 wherein said hardware gate is locked by said switch system base instruction when executing and unlocked by said switch system base instruction when it terminates execution.